Scalable Look-Ahead Linear Regression Trees

David S. Vogel
A.I. Insight, Inc.
Orlando, FL
dvogel@aiinsight.com

Ognian Asparouhov
A.I. Insight, Inc.
Orlando, FL
oasparouhov@medai.com

Tobias Scheffer
Max Planck Institute for Computer Science, Saarbruecken, Germany
scheffer@mpi-inf.mpg.de

ABSTRACT
Most decision tree algorithms base their splitting decisions on a piecewise constant model. Often these splitting algorithms are extrapolated to trees with non-constant models at the leaf nodes. The motivation behind Look-ahead Linear Regression Trees (LLRT) is that out of all the methods proposed to date, there has been no scalable approach to exhaustively evaluate all possible models in the leaf nodes in order to obtain an optimal split. Using several optimizations, LLRT is able to generate and evaluate thousands of linear regression models per second. This allows for a near-exhaustive evaluation of all possible splits in a node, based on the quality of fit of linear regression models in the resulting branches. We decompose the calculation of the Residual Sum of Squares in such a way that a large part of it is pre-computed. The resulting method is highly scalable. We observe it to obtain high predictive accuracy for problems with strong mutual dependencies between attributes. We report on experiments with two simulated and seven real data sets.

Categories and Subject Descriptors
I.5.1 [**Pattern Recognition**]: Models – *Statistical.*

General Terms
Algorithms.

Keywords
Model Tree, Linear Regression Tree, scalable algorithms, regression.

1. INTRODUCTION
Many regression problems cannot adequately be solved by a single regression model. Decision tree algorithms provide the basis for recursively partitioning the data, and fitting local models in the leaves. The CART decision tree algorithm [2] selects the split variable and split value that minimizes the weighted sum of the variances of the target values in the two subsets. Identically, the CART algorithm finds a split that minimizes the Residual Sum of Squares (RSS) of the model when predicting a constant value (the mean) for each of the subsets. In contrast to Linear Regression Trees [12], the CART regression tree algorithm uses the mean for the target variable as the prediction for each of the subsets. Therefore, the CART splitting criterion is appropriate for a piecewise constant model.

KDD'07, August 12–15, 2007, San Jose, California, USA.

Multiple authors have proposed methods to improve upon the accuracy of regression trees by generating models in the leaf nodes of the tree rather than to simply predict the mean. M5 [12], HTL [14], SECRET [5], incremental learning [11], SUPPORT [3], GUIDE [8], PHDRT [7] and SMOTI [9] are some of the model tree algorithms that have been proposed. We refer to the term "model tree" as the more general form of Linear Regression Trees, where the models fit at each leaf node of the partitioned data set may be any predictive model; not necessarily a linear regression model. However, these algorithms differ from each other primarily in their criteria for splitting the data.

LLRT performs a near-exhaustive evaluation of the set of possible splits in a node; splits are evaluated with respect to the quality of the linear regression models, fitted to the instances in the resulting branches. While this algorithm can be as time consuming as other model tree algorithms such as M5, LLRT is sufficiently scalable to process very large data sets. Since it is more robust than many other tree-based algorithms, we observe it to obtain higher predictive accuracy for all data sets we examined with strong mutual dependencies between attributes.

The rest of this paper is structured as followed. We review related work in Section 2. We introduce terminology in Section 3 and present the LLRT in Section 4. Section 5 discusses experimental results, Section 6 concludes.

2. RELATED WORK
Quinlan [12] presents the M5 algorithm which uses a splitting criterion similar to that used by CART. While CART minimizes the weighted sum of the *variances* of the target values in the two subsets, M5 minimizes the weighted sum of the *standard deviations*. The idea is similar and yields similar tree structures. The main difference between M5 and CART is that M5 gains increased accuracy by fitting a linear regression model in each of the leaves of the tree, instead of using the mean value of the target.

Torgo presents the HTL algorithm [14] which uses the CART algorithm to determine the partitions. One key difference between HTL and M5 is that HTL is not restricted to linear models within the leaf nodes. It is more of a clustering approach where each leaf node is evaluated for the appropriate model type.

Dobra and Gehrke present SECRET [5] which demonstrates that using a classification approach to partitioning the data may be closer to a globally optimal partition than the basic CART optimization criterion.

The principle of fitting models to the branches of candidate splits is not new. For more than a decade, Karalic's RETIS [6] model tree algorithm optimizes the overall RSS, simultaneously optimizing the split and the models in the leaf nodes to minimize the global RSS. However, this algorithm is often cited as being intractable, and non-scalable because of the huge complexity of this optimization [11][10][5]. In Karalic's paper, the largest sample size used for experimentation is 300 examples; this provides little evidence to support the algorithm's scalability.

Most papers cite RETIS as a non-practical approach and suggest more scalable algorithms that do not optimize the overall RSS directly, but try to get a little bit closer to the globally optimal split than the predecessor algorithms. SUPPORT [3] and GUIDE [8] are such kinds of algorithms that split the data set based on the distribution of the residuals of a local linear regression model. These algorithms do not optimize globally, but capture the idea of effective clustering by generating partitions of data that differ from one-another.

A more recent innovation in model tree algorithms is SMOTI [9] which allows regression models to exist in the upper parts of the tree instead of only the leaves. This allows for individual predictors to have both global and local effects on the model tree. The SMOTI split evaluation criterion is based on the regression node *above* the node being split whereas to truly obtain a global model an algorithm must examine the predictions specific to the child nodes. The latter is much more intensive computationally because the model for the prediction conditional upon the split changes for each possible split value.

Potts and Sammut [11] discusses an algorithm that bases its splitting decisions on the performance of linear models fitted to the branches of candidate splits. In order to make this approach scalable, only a constant number of candidate splits is considered for each attribute. Potts and Sammut also present an online version of their algorithm.

Similar to this paper, Torgo [13] addresses the issue of efficiently evaluating the candidate splits of a linear regression tree. However, the paper briefly describes the problem setting and formulation and does not provide implementation ideas nor experimental results.

3. PROBLEM SETTING

We study the problem of learning a binary regression tree $f : x \mapsto y$ from data, such that the RSS, $\sum_{i=1}^{N} (f(x_i) - y_i)^2$, is minimized.

Given N training examples and *P* features (also called predictors), define $X \in \Re^{Nx(P+1)}$ whose *N* rows are given by $[1, x_i] \in \Re^{P+1}$, and $y \in \Re^{Nx1}$ containing the corresponding real value labels. The CART approach to generating splits in a regression tree is to minimize the RSS defined by

$$\text{RSS} = \sum_{i \in I_L} (y_i - \bar{y}_L)^2 + \sum_{i \in I_R} (y_i - \bar{y}_R)^2 . \qquad (1)$$

In (1), $\bar{y}_L$ and $\bar{y}_R$ are the means of the target variable in the left and right partitions, I_L and I_R are the set of indices within the left and right partitions, respectively. It is not difficult to construct an algorithm to exhaustively search all possible split variables and split values to obtain the best partition for optimizing (1). This optimization is appropriate for a regression tree, being that the mean values of the left and right nodes will be used as predictions.

With linear regression trees, the appropriate error function would be the same as (1), except that the mean values would be replaced by the models within the leaf nodes. This would minimize the more global measure of RSS,

$$\text{RSS} = \sum_{i \in I_L} (y_i - \hat{f}_L(x_i))^2 + \sum_{i \in I_R} (y_i - \hat{f}_R(x_i))^2 , \qquad (2)$$

which takes into consideration the optimal models within each side of the split, where $\hat{f}_L$ and $\hat{f}_R$ represent the optimal model predictions in the left and right partitions, respectively.

The optimal models $\hat{f}_L$ and $\hat{f}_R$ change for every possible split value of every possible split variable. Due to the complexity of optimizing (2), most model tree algorithms that have been proposed optimize locally, not necessarily using the identical statistic as in (1), but using the same idea of using local optimization as an estimate for optimizing a global model tree.

An important assumption throughout this paper is that $P<<N$, where the applicable data set may be massive in terms of the number of observations *N*, but relatively small in terms of the number of predictors *P*. Due to the large number of matrix solutions that are necessary, the complexity of LLRT is approximately $O(P^5N)$. For massive datasets that can be pre-processed down to 120 attributes or less, this algorithm will perform well. The algorithm requires a single $O(P^2N)$ scan for evaluating any number of split values for a given split variable. This is equivalent to running a linear regression in addition to $O(P^4)$ operations for the stepwise regression process at each potential split.

4. MODEL TREE ALGORITHM

First we discuss the calculation of RSS for regression models in a *k*-fold cross validation setting so that models may be evaluated quickly. Normally RSS is calculated by first applying the model and then summing up the squares of the residuals. This requires an $O(P \cdot N)$ operation. We will derive an $O(P^2)$ operation that accomplishes the same calculation without scanning the data set. The second part of the description explains how the RSS calculations are applied within the algorithm to be able to optimize Equation (2) in the splitting process.

4.1 RSS Calculations

The linear regression equation within a given leaf of the model tree is $\hat{Y} = Xc$, where $c \in \Re^{(P+1)x1}$ is the regression coefficient vector. In matrix notation, the RSS can be written as $RSS = \|Xc - Y\|^2$. The regression coefficient vector has to be estimated such that it minimizes this RSS: $c^* = \arg\min_c \|Xc - Y\|^2$. In Equation (3), we rephrase the RSS using elementary matrix algebra.

$$\begin{aligned} \text{RSS} &= \|Xc - Y\|^2 \\ &= \langle Xc - Y, Xc - Y \rangle \\ &= c^T X^T Xc - Y^T Xc - c^T X^T Y + Y^T Y \\ &= c^T Rc - 2c^T b + Y^T Y \end{aligned} \qquad (3)$$

where $R = X^T X$ and $b = X^T Y$.

One can minimize the RSS by setting the derivative

$$\frac{\partial RSS}{\partial c} = \frac{\partial (c^T X^T Xc - 2c^T X^T Y + Y^T Y)}{\partial c}$$
$$= 2X^T Xc - 2X^T Y$$

to zero, which gives $Rc=b$. Traditionally in regression models c is solved by using Singular Value Decomposition to obtain the inverse of R:

$$c = R^{-1} b = (X^T X)^{-1} X^T Y \qquad (4)$$

Normally the execution time for this part of the modeling process is insignificant. However, in the LLRT algorithm, there will be millions of mini-regression models that need to be created. Therefore it is imperative that a significant portion of the calculations is carried out with a more efficient algorithm. There are quicker alternatives to Singular Value Decomposition[4] such as G.E. (Gaussian Elimination) or Cholesky Decomposition if the matrix R is Singular Positive Definite. We use G.E. with full pivoting. One problem that needs to be addressed when using G.E. is that it will not work with singular matrices. For use of G.E. in regression problems, we use an implementation that carefully detects and removes singularities from the matrix R and sets the corresponding coefficients to zero. Using G.E., the full inverse of R needs not be calculated. G.E. can solve directly for the coefficient matrix c.

The key to avoiding over-fitting in a model so complex as a linear regression tree is to incorporate sampling methods. We use S-fold validation because it works well within the given formulation. We define X and Y as above and separate the N examples of the data within each tree node into S partitions. For these partitions we define $X_1 \ldots X_S$ and $Y_1 \ldots Y_S$. Now, the regression coefficient vector corresponding to a partition k is defined by $c_k = \arg\min_c \|(X - X_k)c - (Y - Y_k)\|^2$ where $X - X_k$ contains all examples not included in the k^{th} fold of the data. This way, the model for a given partition is out-of-sample for that partition. The RSS of c_k regarding partition k is

$$\begin{aligned} RSS_k &= \|X_k c_k - Y_k\|^2 \\ &= c_k{}^T R_k c_k - 2c_k{}^T b_k + Y_k{}^T Y_k \end{aligned}$$

where $R_k = X_k{}^T X_k$ and $b_k = X_k{}^T Y_k$. The total RSS is now given as

$$\begin{aligned} RSS &= \sum_{k=1}^{S} RSS_k \\ &= \sum_{k=1}^{S} c_k{}^T (R_k c_k - 2b_k) + Y^T Y . \end{aligned} \qquad (5)$$

4.2 Evaluating the Split Candidates

At each potential split value of each possible variable, the RSS must be calculated for the optimal model. In order to estimate the optimal model, a forward stepwise regression model is generated in the prospective child nodes. We use k-fold validation to determine the variables to be added into each of the two models. For each prospective variable being added to the regression model, the matrices need not be reconstructed because they are subsets of $R_{1..S}$ and $b_{1..S}$ that are pre-calculated for all the features in the model. For each potential feature, c must be solved, which is an operation of $O(P^3)$.

The RSS for each model is calculated based on (5). This avoids the $O(S \cdot P \cdot N)$ operations of applying each model and then summing up the squares of the residuals. Using (5), the evaluation of the model only requires $O(S \cdot P^2)$ operations, which is independent of the large dimension N.

Therefore, at a given prospective split point, the evaluation of RSS (based on Equation 2) is independent of N. This is remarkable because several models need to be created and evaluated to determine the optimal selection of variables for each node. Furthermore, these models are all evaluated out-of-sample using k-fold validation. Because the number of models tried during variable selection is $O(P)$ and each model requires $O(P^3)$ operations (done S times for validation), the total number of operations needed to evaluate a given split is $O(S \cdot P^4)$. As mentioned in the introduction, this is fairly quick given that P is relatively small (in our targeted applications, below 120).

4.3 Optimizations

It was mentioned earlier that LLRT does a *nearly* exhaustive search of the possible split values. Due to the high number of calculations required to evaluate a given split, we find it to be unnecessary to evaluate *every* possible split value. If a value is continuous and has a different value for every record, it is not likely to make a big difference in the overall performance if the split value shifts a few observations away from the optimal value. We conjecture that it is sufficient to try 10 or 20 possible split values. This way each of the candidate values would be close enough together that LLRT would detect a split very close to the optimal split value, and it would eliminate another factor of N in the order of magnitude of the calculation.

Instead of re-calculating $R_{1..S}$ and $b_{1..S}$ at each possible split value ($O(P^2 \cdot N)$ and $O(P \cdot N)$, respectively), LLRT employs a technique similar to that described by Natarajan et. al. [10] which allows the sufficient statistics to be *updated* instead of re-calculated. Recognizing that $R_{1..S}$ and $b_{1..S}$ are additive matrices, these values may be calculated for subsets of observations (in between the split values) and subsequently added to one leaf node and subtracted from the other leaf node. The effect is that instead of calculating the entire matrices 10 or 20 times, they only need to be calculated once (one piece at a time). The net result is that the aggregate of these updates totals a single $O(P^2 \cdot N)$ operation for a given split variable regardless of how many split values are evaluated.

4.4 Stopping Rules

It must be taken into consideration that the algorithm uses the same samples throughout the entire evaluation of an optimal split. Therefore it is possible to over-fit the out-of-sample result because the same samples are used repeatedly. Many stopping criteria for decision tree learning have been explored. We use the following heuristic: the algorithm performs an extra check on the proposed "best split". Based on the apparently best split, the algorithm re-samples the data and re-evaluates the combined accuracy of the child nodes. If the accuracy again supersedes the accuracy of the parent node, then the split is made. If not, then the split is denied and a stepwise regression is performed as the final model for that node.

4.5 Algorithm

The LLRT algorithm is described in Table 1.

Table 1: LLRT model tree algorithm.

function Split(leaf t, data matrix X, target vector Y)

Pre-calculate $R_{1..S}$ and $b_{1..S}$

determine model minimizing RSS, using (5), using a stepwise regression, and using subsets of $R_{1..S}$ and $b_{1..S}$

Initialize BestRSS = RSS for the stepwise regression

for each split variable i

create sort index on i^{th} variable to be used to scan observations

initialize $R_{1L}..R_{SL} = 0$, $b_{1L}..b_{SL} = 0$ (no data in Left Split)

initialize $R_{1R}..R_{SR} = R_1..R_S$, $b_{1R}..b_{SR} = b_1..b_S$ (all data in Right Split)

for each of q-1 possible partitions j

scan the j^{th} group of N/q observations to calculate:

$R\_Sub_1..R\_Sub_S$, $b\_Sub_1..b\_Sub_S$

Add all "Sub" matrices to $R_{1L}..R_{SL}$, $b_{1L}..b_{SL}$

Subtract all "Sub" matrices from $R_{1R}..R_{SR}$, $b_{1R}..b_{SR}$

Use stepwise regression to minimize RSS_L

Use stepwise regression to minimize RSS_R

if $RSS_L + RSS_R <$ BestRSS

BestRSS = $RSS_L + RSS_R$

Mark this split as the best split

end if

end for

end for

if BestRSS not improved by any split

Break (this becomes a leaf node)

else

partition data based on best split

Split(t_L, X_L, Y_L) (recursively split left data)

Split(t_R, X_R, Y_R) (recursively split right data)

end if

4.6 Execution Time

There are two elements to the complexity of the calculations:

(1) the model evaluations at each possible split value;

(2) the data scans for modifying $R_{1..S}$ and $b_{1..S}$.

In section 4.2 it was explained that the model evaluations for each possible split have complexity $O(S \cdot P^4)$, where S is the number of folds in the k-fold validation and P is the number of predictors. In section 4.3 it was shown that the data scans for each split variable have complexity $O(P^2 \cdot N)$. The total complexity per split variable is $O(P^2(cqSP^2+N))$, where c is a constant and q is the number of split values evaluated per variable and the number of folds in the k-fold validation.

Multiply this by P possible split variables and the total complexity for determining the best possible split for a given data node is $O(P^3(cqSP^2+N))$.

In order to build the entire model tree, the algorithm can be shown to require at most

$$O\left[P^3 N \left(\frac{cqSP^2}{d} + \log_2(N/d) \right) \right] \qquad (6)$$

operations, where d is the average number of observations in a leaf node.

In short, the algorithm is $O(P^5)$ in the worst case with respect to the number of features and approaches $O(P^3)$ for very large data sets. The algorithm is $O(N)$ with respect to the number of observations, and approaches $O(N \log N)$ for very large data sets.

5. EMPIRICAL RESULTS

In this section, we study the predictive accuracy and execution time of the LLRT algorithm in comparison to other methods. Since linear regression trees are a hybrid between regression trees and linear regression, we use a stepwise linear regression, an M5 decision tree and a REP decision tree as three of the comparative techniques. Additionally, we compare against the M5 linear regression tree. All four comparative algorithms are run using the implementations in WEKA [15], software that is publicly available. We study five problems that exhibit strong mutual dependencies between attributes.

We first study two artificial problems designed to help understand the kinds of patterns that can be found using the look-ahead approach. The remaining three problems contain real world data sets from three very different areas: Customer relationship management, particle physics and several risk assessment models. These data sets are not enormous, but are large enough to study the scalability of LLRT in comparison to other commonly used algorithms. Mean Squared Error (MSE) is our measure of accuracy since it is proportional to RSS (MSE = RSS / *N*), the objective function that is optimized by the methods being compared.

5.1 Simulated Data Set #1

The first simulated data set is one that is known to be perfectly predictable with regression lines on subsets of the data resulting from a single partition. The equation is

$$y = \begin{cases} 0, & -4 \le x < -2 \\ 0.5 + 0.25 * x, & -2 \le x \le 2 \end{cases} \quad (7)$$

with x uniformly distributed between -4 and 2. The graph is illustrated in Figure 1.

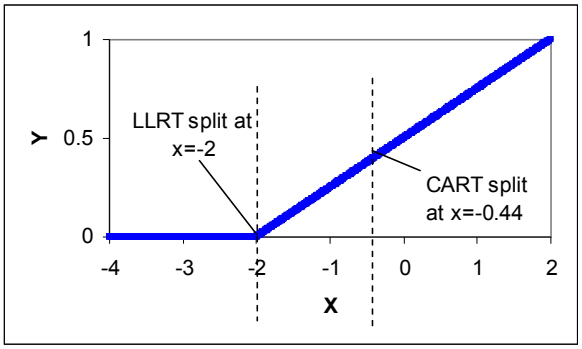

Figure 1: The relationship between variables X and Y for the first simulated data set.

A model tree using the CART approach partitions this data set at the point x=-0.44 minimizing the combined RSS of both partitions using the *means* of each subset as the prediction. LLRT partitions the data at the point x=-2 because it is able to look one step ahead to determine that the best global RSS for a piecewise linear model (in this case, a perfect model) can be obtained from this partitioning. Both algorithms make plausible decisions; CART optimizes for a piecewise constant model and LLRT optimizes for a piecewise linear model. However, this illustrates why algorithms employing the CART split selection for induction of linear regression trees generally do not find the best partitions.

5.2 Simulated Data Set #2

The second simulated data set is a multivariate scenario where many traditional approaches will fail to efficiently fit the underlying patterns. X_1 is a binary attribute (50% of values are 0, 50% of values are 1), and X_2 is a uniformly distributed continuous attribute [-1,1]. There are 500 observations in the training set and 500 observations in the validation set. The equation for the underlying model is

$$y = \begin{cases} 3x_2, & x_1 = 0 \\ -x_2, & x_1 = 1 \end{cases} \quad (8)$$

To make the simulated data set more realistic, uniformly distributed random noise was added to the y-values; graphs are illustrated in Figures 2 and 3.

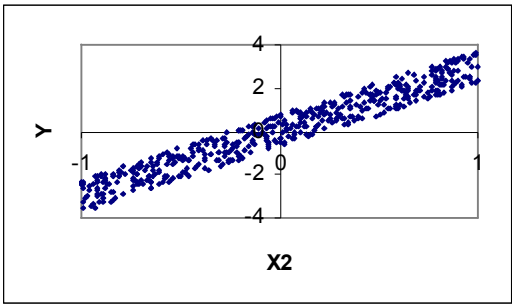

Figure 2. The scatter plot of X_2 versus y when X_1=0

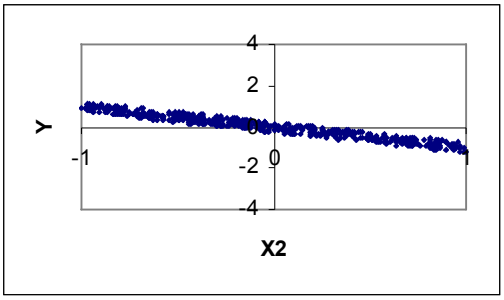

Figure 3. The scatter plot of x_2 versus y when x_1=1

Table 2 compares the results for linear regression versus two linear regression trees, the first split by x_2 (selected by CART as the splitting variable) and the second tree split by x_1 (selected by LLRT as the splitting variable).

Table 2. Model MSE comparison for the second generated data set.

	Regression	**CART split on X_2**	**LLRT split on X_1**	**True Model**
Training	1.338	0.440	0.104	0.103
Validation	1.491	0.490	0.101	0.101

The "true model" is the best possible model for this data which is defined by Equation (8) which was used to generated the data. With most real world data sets, one would not expect to see such a pronounced difference between the linear regression and LLRT. This data was generated as an extreme example for the purpose of illustrating the type of scenario where a model tree with "look-

ahead" capabilities would be beneficial. For this particular data set, the mean value of y is nearly identical for both subsets of the data split on x_1. Therefore, a conventional decision tree algorithm would select the partition on x_2 instead of x_1. It is clear from the results that a partition on x_2 is by far inferior to a partition on x_1. The LLRT algorithm detects that a partition on x_1 gives the maximum benefit to the overall model when using regression models within each of the partitions, and makes the correct choice in using x_1 as the splitting variable.

5.3 Customer Relationship Management

The KDD-cup 98 competition data comes from PVA (Paralyzed Veterans of America) and is based on a fund-raising effort through a series of mailings to prospective donors (http://www.kdnuggets.com/meetings/kdd98/kdd-cup-98.html). The models were evaluated based on a measure of profitability of the fund-raiser when using the given model (donation minus cost of mailing for potential donors selected by the model). The model was constructed on the training data set, consisting of 95,413 observations with 481 fields. Some additional fields were created based on sensible transformation of existing fields in the data set and variable selection was used to extract a subset of 85 features to be used as predictors. The validation set contained 96,367 observations, and was used to evaluate the model. Table 3 compares LLRT and M5 results to the top 3 results in the competition.

Table 3. Model comparison for the KDD Cup 1998 data set.

Model	Profit
LLRT	$15,182
M5	$12,225
KDD Cup 1st Place	$14,712
KDD Cup 2nd Place	$14,662
KDD Cup 3rd Place	$13,954

Even though LLRT is compared to winning algorithms from several years ago, the result is significant because it is automated and performs variable selection within the algorithm. Also notable is the comparison to the accuracy cited in [10], where the model tree returns a profit of $13,778. The linear regression tree of Natarajan et. al. out-performs M5 but does not benefit from the look-ahead approach of LLRT. LLRT is responsible for nearly a 25% increase in profit over the model generated by M5, and in this case the difference between the gold medal and no medal.

5.4 Particle Physics Data

The KDD-cup 2004 competition data comes from SLAC (Stanford Linear Accelerator) and contains 78 properties of 150,000 particles (http://kodiak.cs.cornell.edu/kddcup/). Since the task for the competition was a classification task (differentiating matter from anti-matter), we use the 24th column in this data set as the continuous target for our experiments.

LLRT was compared to three algorithms: Linear Regression (stepwise), the M5 decision tree, and the M5 linear regression tree. Samples of sizes 5,000, 10,000, 20,000, 40,000, and 100,000 observations were randomly extracted from the complete data file to be used as training sets. A subset (disjoint from the training samples) of 10,000 observations was removed from the data and used as a validation data set. Having training sets of different sizes allows us to observe the improved performance of LLRT as the training size increases (Figure 4).

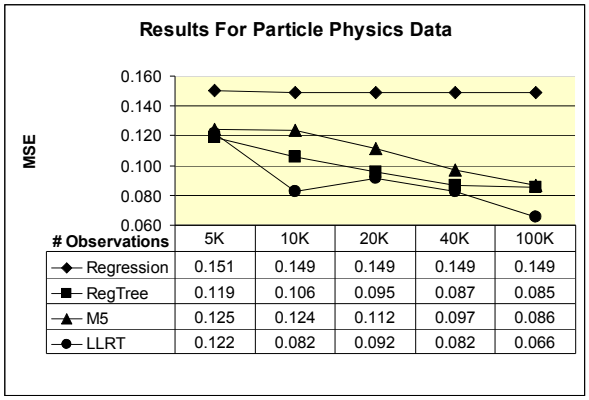

Figure 4: Results for Particle Physics Data. Validation set MSE is plotted against models from various training set sizes.

Figure 4 demonstrates how the linear regression does not improve as the sample size increases because the technique is incapable of using more data to make more complex models. M5 and the regression tree improve substantially as the size of the data set increases. However, neither of these algorithms improves nearly as much as LLRT, whose MSE drops by 45% as the training set size increases from 5,000 to 100,000 observations. As the size of the training set reaches 100,000 observations LLRT outperforms M5 and the regression tree by over 20% as its validation set RSS shrinks to 0.066. This particular data set contains variables with very complex relationships to one-another. This is why the difference in accuracy is so pronounced between LLRT and M5. As the relationships in a data set become more complex, it becomes increasingly important to have look-ahead capabilities. Simple linear approaches do not perform well, and variance-reducing partitioning procedures capture only some of these relationships.

Figures 5 and 6 graph the execution times for M5 and LLRT.

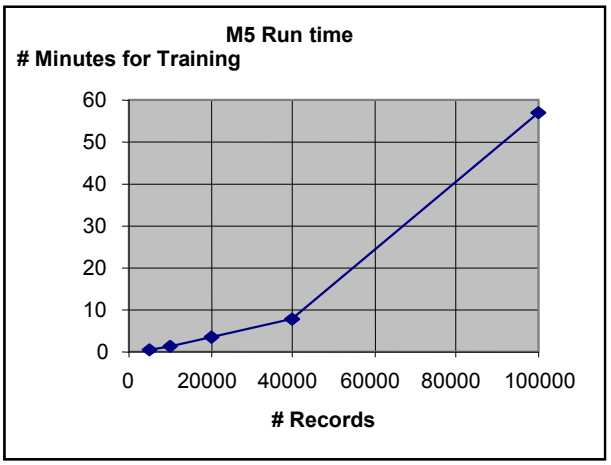

Figure 5: Execution time for M5 on the particle physics data.

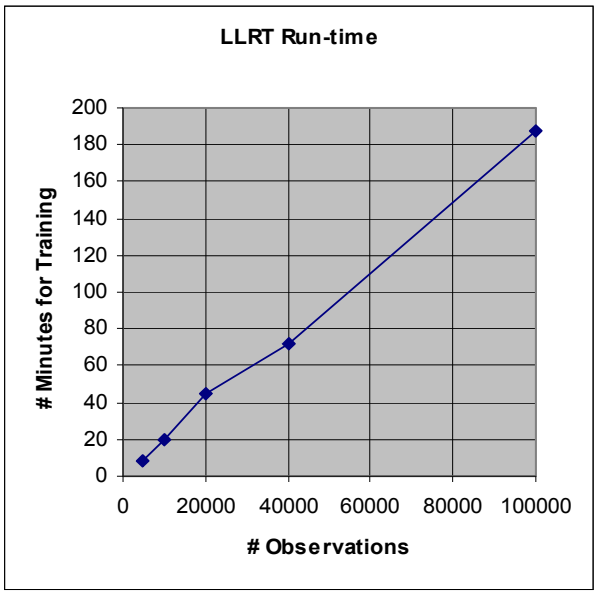

Figure 6: Execution time for LLRT on the particle physics data. Execution time is approximately linear in the sample size.

For this particular data set, LLRT is slower than M5. This is the one drawback of a more exhaustive search for globally optimal splits, despite the numerous optimizations. However, it should be noted that LLRT run-time increases approximately linearly with respect to the sample size. We know from (6) that the LLRT is really O(N log N) for large data sets, but in this case N is not large enough for the "log N" factor to be noticeable. M5 is less scalable because it appears from our experiments that its execution time increases much more rapidly than O(N log N) for the larger data sets. For a training set size of 5,000 observations, M5 is 16 times faster than LLRT. For a training set size of 100,000 observations, M5 is only 3 times faster than LLRT. However, the quality of the partitionings obtained by LLRT is substantially higher yielding an accuracy improvement of more than 20%!

5.5 Health Risk Assessment

There are several predictive modeling problems in the area of health risk assessment. The attributes include demographic information as well as a variety of medical conditions.

The first problem we discuss is to predict the overall costs that health insurance customers will incur in the next 12 months [1]. The target variable is the percentile (0-100) of total cost associated with the second year of data for the members in the database. The risk assessment data set comes from MEDai's RNC (Risk Navigator Clinical) database.

The second problem is to predict the total count of drugs incurred by health plan members in the following 12 months.

The third problem is a concurrent risk model, predicting a measure of risk based on various diagnoses and demographic factors.

The fourth and fifth problems are models predicting LOS (length of stay) in a hospital for psychosis and dermatology patients, respectively.

In the medical industry, comprehensible models are imperative and managers desire to be provided not only with a prediction for each instance, but also with an explanation. Linear regression trees are therefore much more adequate than, for instance, kernel machines or other, more complex statistical regressors.

Table 4. Model comparison across the five medical data sets for Linear Regression, REP Decision Tree, M5 Decision Tree, M5 LRT and LLRT.

Data:	Cost	Drug	Risk	Psych	Derm
Training N	30810	109248	212104	111922	100003
Test N	10301	12154	53485	16058	14246
Pred-ictors	58	53	17	61	62
# Rules					
LinReg	1	1	1	1	1
REP	367	437	520	291	389
M5 T	85	221	322	560	204
M5 LRT	6	67	115	456	131
LLRT	9	11	46	14	8
Minutes					
LinReg	1:10	3:20	0:10	3:16	2:55
REP	0:12	0:55	0:47	1:11	0:49
M5 T	9:30	6:08	4:20	6:37	3:16
M5 LRT	16:48	39:24	4:45	17:46	12:08
LLRT	4:13	7:49	5:00	17:44	13:48
MSE					

LinReg	502.4	19.82	10.75	2403	1925
REP	478.2	19.20	9.682	1537	1148
M5 T	458.7	19.06	9.528	1938	1367
M5 LRT	448.1	18.98	9.421	1519	1086
LLRT	**441.7**	**18.49**	**8.879**	**1469***	**1064****

*Increasing the training set size of Psych from 111922 to 225025 lowered the test set error to 1424, 3% additional improvement. M5 was not sufficiently scalable to run this comparison. LLRT generated 22 rules in 51:49.

**Increasing the training set size of Derm from 100003 to 228639 lowered the test set error to 1035, 3% additional improvement. M5 was not sufficiently scalable to run this comparison. LLRT generated 25 rules in 45:22.

In all cases, LLRT outperformed M5 by 2-6%. It is important to note that the training sets used for comparison were in two cases reduced in size (taking random subsets) in order for M5 to be capable of running a comparative model. LLRT's scalability allows us to run the entire training sets and in both cases further reduced the MSE by an additional 3%. While the current implementation of LLRT is constrained to the amount of RAM on a computer, the techniques used by Natarajan et al [10] may be applied to this algorithm to allow it to train on considerably more massive databases.

6. CONCLUSIONS

Tree building algorithms assess attributes independently of other attributes, thereby failing to pay sufficient attention to interactions. Tree-based algorithms such as CART have been known to find interactions in data, but do not focus on these interactions in the determination of the splitting variables to find a globally optimal model. We developed a method that efficiently calculates the k-fold cross validation RSS of millions of regression models by pre-computing the data-dependent part of the matrix operations prior to iterating over the possible splits in the current data. This allows us to efficiently evaluate splits by evaluating the regression models that result from them.

For small samples, the LLRT algorithm is slower by a modest factor than conventional methods such as M5 or regression trees; however, it scales favorably to large scale regression problems even though it exercises a substantially broader search.

Experimental results show that LLRT outperforms less robust learning methods for regression problems that are characterized by strong mutual interactions between attributes.

Health risk assessment is in itself a significant application characterized by interacting features and large training databases. The performance of LLRT for this problem justifies our research. For the customer relationship problem of the KDD Cup 1998, LLRT outperformed the winning competition entries, even though it is an "out of the box" method and the winning solutions were carefully tailored for this problem. For the particle physics problem LLRT again outperformed M5, decision trees and linear regression.

7. REFERENCES

[1] Axelrod, R. & Vogel, D. (2003). Predictive Modeling in Health Plans. *Disease Management and Health Outcomes*, 11 (12): 779-787.

[2] Breiman, L., Friedman, J., Olshen, R., Stone, C. *Classification and Regression Trees*, Wadsworth, Belmont, CA, 1984.

[3] Chaudhuri, P., Huang, M., Loh, W., and Yao, R. Piecewise Polynomial Regression Trees, *Statistica Sinica*, vol. 4, pp. 143-167, 1994.

[4] Deuflhard, P. and Hohmann, A. *Numerical Analysis in Modern Scientific Computing: An Introduction.* Second revised and extended edition. Texts in Applied Mathematics 43, Springer, 2003.

[5] Dobra, A., and Gehrke, J. SECRET: A Scalable Linear Regression Tree Algorithm. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Edmonton, Alberta, Canada, July 2002).

[6] Karalic, A. Linear Regression in Regression Tree Leaves. In *International School for Synthesis of Expert Knowledge*, Bled, Slovenia, 1992.

[7] 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.

[8] Loh, W. Regression Trees with Unbiased Variable Selection and Interaction Detection, *Statistica Sinica*, vol. 12, pp. 361-386, 2002.

[9] Malerba, D., Appice, A., Ceci, M., and Monopoli, M. Trading-Off Local versus Global Effects of Regression Nodes in Model Trees. In *Proceedings of Foundations of Intelligent Systems, 13th Int'l Symp*., pp. 393-402 (2002).

[10] Natarajan, R., and Pednault, E. Segmented Regression Estimators for Massive Data Sets. In *Proceedings of the SIAM International Conference on Data Mining (SDM '02)* (Chicago, IL, April 11-13, 2002).

[11] Potts, D. and Sammut, C. Incremental Learning of Linear Model Trees. In *Machine Learning,* 61, 5-48, 2005.

[12] Quinlan, J.R. Learning with Continuous Classes. In *5th Australian Joint Conference on Artificial Intelligence*, pp. 343-348 (1992).

[13] Torgo, L. Computationally Efficient Linear Regression Trees. In *Classification, Clustering and Data Analysis: Recent Advances and Applications* (Proceedings of IFCS), 2002.

[14] Torgo, L. Functional Models for Regression Tree Leaves. In *Proceedings of the 14th International Conference on Machine Learning* (ICML-97). pp. 385-393. Morgan Kaufmann (1997).

[15] Witten, I.H. and Frank, Eibe. *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd Edition, Morgan Kaufmann, San Francisco, CA, 2005.