SECRET: A Scalable Linear Regression Tree Algorithm

Alin Dobra
Department of Computer Science
Cornell University
Ithaca, NY 14853
dobra@cs.cornell.edu

Johannes Gehrke
Department of Computer Science
Cornell University
Ithaca, NY 14853
johannes@cs.cornell.edu

ABSTRACT

Recently there has been an increasing interest in developing regression models for large datasets that are both accurate and easy to interpret. Regressors that have these properties are regression trees with linear models in the leaves, but so far, the algorithms proposed for constructing them are not scalable. In this paper we propose a novel regression tree construction algorithm that is both accurate and can truly scale to very large datasets. The main idea is, for every intermediate node, to use the EM algorithm for Gaussian mixtures to find two clusters in the data and to locally transform the regression problem into a classification problem based on closeness to these clusters. Goodness of split measures, like the gini gain, can then be used to determine the split variable and the split point much like in classification tree construction. Scalability of the algorithm can be enhanced by employing scalable versions of the EM and the classification tree construction algorithms. Tests on real and artificial data show that the proposed algorithm has accuracy comparable to other linear regression tree algorithms but requires orders of magnitude less computation time for large datasets.

1. INTRODUCTION

Regression is a very important data mining problem. One very important class of regression models is regression trees. Even though they were introduced early in the development of classification trees (CART, Breiman et al. [4]), regression trees received far less attention from the research community. Quinlan [16] generalized the regression trees in CART by using a linear model in the leaves to improve the accuracy of the prediction. The impurity measure used to choose the split variable and the split point was the standard deviation of the predictor for the training examples at the node. Karalilc [11] argued that the mean square error of the linear model in a node is a more appropriate impurity measure for the linear regression trees since data well predicted by a linear model can have large variance. This is a crucial observation since evaluating the variance is much easier than estimating the error of a linear model (which requires solving a linear system). Even more, if discrete attributes are present among the predictor attributes and binary trees are built (as is the case in CART), the problem of finding the best split attribute becomes intractable for linear regression trees since the theorem that justifies a linear algorithm for finding the best split (Theorem 9.4 in [4]) does not seem to apply. To address computational concerns of normal linear regression models, Alexander and Scott [1] proposed the use of simple linear regressors (i.e., the linear model depends on only one predictor attribute), which can be trained more efficiently but are not as accurate.

Torgo proposed the use of even more sophisticated functional models in the leaves (i.e., kernel regressors) [19, 18]. For such regression trees both construction and deployment of the model is expensive but they potentially are superior to the linear regression trees in terms of accuracy. More recently, Li et al. [12] proposed a linear regression tree algorithm that can produce oblique splits[1] using Principal Hessian Analysis but the algorithm cannot accommodate discrete attributes.

There are a number of contributions coming from the statistics community. Chaudhuri et al. [5] proposed the use of statistical tests for split variable selection instead of error of fit methods. The main idea is to fit a model (constant, linear or higher order polynomial) for every node in the tree and to partition the data at each node into two classes: datapoints with positive residuals[2] and datapoints with negative residuals. In this manner the regression problem is locally reduced to a classification problem, so it becomes much simpler. Statistical tests used in classification tree construction, Student's t-test in this case, can be used from this point on. Unfortunately, it is not clear why differences in the distributions of the signs of the residuals are good criteria on which decisions about splits are made. A further enhancement was proposed recently by Loh [13]. It consists mostly in the use of the χ^2-test instead of the t-test in order to accommodate discrete attributes, the detection of interactions of pairs of predictor attributes, and a sophisticated calibration mechanism to ensure the unbiasedness of the split attribute selection criterion.

In this paper we introduce SECRET (Scalable EM and Classification based Regression Trees), a new construction

[1]Oblique splits are linear inequalities involving two or more predictor attributes.
[2]Residuals are the difference between the true value and the value predicted by regression model.

algorithm for regression trees with linear models in the leaves, which produces regression trees with accuracy comparable to the ones produced by existing algorithms and at the same time requiring far less computational effort on large datasets. Our experiments show that SECRET improves the running time of regression tree construction by up to two orders of magnitude compared to previous work while at the same time constructing trees of comparable quality. Our main idea is to use the EM algorithm on the data partition in an intermediate node to determine two Gaussian clusters, hopefully with shapes close to flat disks. We then use these two Guassians to locally transform the regression problem into a classification problem by labeling every datapoint with class label 1 if the probability of belong to the first cluster exceeds the probability of belong to the second cluster, or class label 2 if the converse is true. A split attribute and a corresponding split point to seperate the two classes can be determined then using goodness of split measures for classification trees like the gini gain [4]. Least square linear regression can be used to determine the linear regressors in the leaves.

The local reduction to a classification problem allows us to avoid forming and solving the large number of linear systems of equations required for an exhaustive search method such as the method used by RETIS [11]. Even more, scalable versions of the EM algorithm for Gaussian mixtures [3] and classification tree construction [9] can be used to improve the scalability of the proposed solution. An extra benefit of the method is the fact that good oblique splits can be easily obtained.

The rest of the paper is organized as follows. In Section 2 we give short introductions to classification and regression tree construction and to the EM algorithm for Gaussian mixtures. In Section 3 we present in greater detail some of the previously proposed solutions and we comment on their shortcomings. Section 4 contains the description of SECRET, our proposal for a linear regression tree algorithm. We show results of an extensive experimental study of SECRET in Section 5 and we conclude in Section 6.

2. PRELIMINARIES

In this section we give a short introduction to classification and regression trees and the EM algorithm for Gaussian mixtures.

2.1 Classification Trees

Classifiers are functional mappings from the crossproduct of the domains of *predictor attributes* $X_1 \ldots X_m$ to the domain of the predicted attribute C. Usually the values of C are called *class labels.*

A special type of classifier is a *classification tree.* A classification tree is a directed, acyclic graph $\mathcal{T}$ with a tree shape. The root of the tree (denoted by $\text{Root}(\mathcal{T})$) does not have any incoming edges. Every other node has exactly one incoming edge and may have 0, 2 or more outgoing edges. We call a node T without outgoing edges a *leaf node*, otherwise T is called an *internal node.* Each leaf node is labeled with one class label; each internal node T is labeled with one predictor attribute X_T, $X_T \in \{X_1, \ldots, X_m\}$ called the *split attribute.* We denote the label of node T by $\text{Label}(T)$.

Each edge (T, T') from an internal node T to one of its children T' has a predicate $q_{(T,T')}$ associated with it where $q_{(T,T')}$ involves only the splitting attribute X_T of node n. The set of predicates Q on the outgoing edges of an internal node T, which we call *splitting predicates* of T, must be non-overlapping and exhaustive.

For a given decision tree $\mathcal{T}$, predictions are made in the usual recursive manner: (1) if the node T is a leaf return $\text{Label}(T)$, (2) if the node T is an intermediate node and for some j $q_{(T,T_j)}$ is true, return the prediction of node T_j.

Given a dataset D, called the set of *training examples*, we would like to build a classification tree that *best* captures the *patterns* in this dataset. Such a classification tree should not only predict well the class label of the training datapoints but also *generalize* to unseen examples that are coming from the same source as the training examples. For this reason a classification tree is usually constructed in two phases [4]. In phase one, the *growth phase*, an overly large classification tree is constructed from the training data. In phase two, the *pruning phase*, the final size of the tree $\mathcal{T}$ is determined with the goal to minimize the error on unseen examples.

Most classification tree construction algorithms grow the tree top-down in the greedy way [4] shown in Figure 1. Note that the algorithm shown in Figure 1 takes a split selection criterion as argument. A number of such split criteria have been proposed in the literature: gini gain, information gain, gain ratio, χ^2-test, G^2-statistic, etc., [15]. We define here only the gini gain since is the only one we use in the present work. The gini gain is based on gini index (introduced by Breiman et al.[4]):

$$gini(T) \stackrel{\text{def}}{=} 1 - \sum_{i=1}^{J} P[i|T]^2$$

The gini gain of a node T when split using the split predicates Q on the predictor attribute X is defined as:

$$GG(T, X, Q) \stackrel{\text{def}}{=} gini(T) - \sum_{j=1}^{n} P[q_{(T,T_j)}(X)|T] \cdot gini(T_j)$$

The gini index is an impurity measure and gini gain represents the gain in purity if the split is made. Choosing split variables X and split predicates Q with greater values of the gini gain result in more progress being made. For the case when there are only two class labels and all the splits are binary, gini gain has a very important property, namely the split predicate can be found in linear time in the size of the domain of the attribute (Theorem 9.4 in [4]). Also, in this situation, gini gain takes the simpler form:

$$GG_b(T, X, Q) = P[1|T]^2 \frac{(P[1|T_1] - P[T_1])^2}{P[T_1](1 - P[T_1])} \qquad (1)$$

A great number of pruning methods have been proposed [15]. In this paper we use Quinlan's resubstitution error pruning [17]. It consists in eliminating subtrees in order to obtain a tree with the smallest error on a separate dataset, called *pruning set.*

2.2 Regression Trees

Regression models or regressors are functional mappings from the cross product of the domains of predictor attributes $X_1, \ldots, X_m$ to the domain of the continuous predicted attribute, Y. They only differ from classifiers in the fact that the predicted attribute is real valued.

Regression Trees, the particular type of regressors we are interested in, are the natural generalization of decision trees

```
Input: node $T$, data-partition $D$, split selection method $\mathcal{V}$
Output: classification tree $\mathcal{T}$ for $D$ rooted at $T$

Top-Down Classification Tree Induction Schema:
BuildTree(node $T$, data-partition $D$,
        split attribute selection method $\mathcal{V}$)
(1)  apply $\mathcal{V}$ to $D$ to find the split attribute $X$ for node $T$
(2)  let $n$ be the number of children of $T$
(2)  if ($T$ splits)
(3)    partition $D$ into $D_1, \ldots, D_n$ and label node $T$
         with split attribute $X$
(4)    create children nodes $T_1, \ldots, T_n$ of $T$ and label
         the edge $(T, T_i)$ with predicate $q_{(T,T_i)}$
(5)    foreach $i \in \{1, .., n\}$
(6)      BuildTree($T_i$, $D_i$, $\mathcal{V}$)
(7)    endforeach
(8)  else
(9)    label $T$ with the majority class label of $D$
(10) endif
```

Figure 1: Classification Tree Induction Schema

for regression problems. Instead of a class label being associated to every node, a real value or a functional dependency of some of the inputs is used to predict the value of the output.

Regression trees in CART have a constant numerical value in the leaves and use the variance as a measure of impurity [4]. Thus the split selection measure for a node T is:

$$\begin{aligned} \mathrm{Err}(T) &\overset{\mathrm{def}}{=} E\left[(Y - E\left[Y|T\right])^2\right] \\ \Delta\mathrm{Err}(T) &= \mathrm{Err}(T) - \sum_{j=1}^{n} P[q_j(X)|T] \cdot \mathrm{Err}(T_j) \end{aligned} \tag{2}$$

The use of variance as the impurity measure is justified by the fact that the best constant predictor in a node is the expected value of the predicted variable given that datapoints belong to the node, $E\left[Y|T\right]$; the variance is thus the mean square error of this best predictor.

As in the case of classification trees, prediction is made by navigating the tree following the branches with predicates that are satisfied until a leaf is reached. The numerical value associated with the leaf is the prediction of the model.

Usually a top-down induction schema algorithm like that in Figure 1 is used also to build regression tress.

Like for classification trees, pruning is used to improve the accuracy on unseen examples. Pruning methods for classification trees can be straightforwardly adapted for regression trees [20].

2.3 The EM Algorithm for Gaussian Mixtures

In this section we give a short introduction to the problem of approximating some unknown distribution, from which a sample is available, with a mixture of Gaussian distributions. The EM algorithm [6] can be used to determine the parameters of the Gaussian distributions of a locally optimal mixture. Our introduction follows mostly the excellent tutorial [2] where details and complete proofs of the EM algorithm for Gaussian mixtures can be found.

The Gaussian mixture density estimation problem is the following: find the most likely values of the parameter set $\Theta = (\alpha_1, \ldots, \alpha_M, \mu_1, \ldots, \mu_M, \Sigma_1, \ldots, \Sigma_M)$ of the probabilistic model:

$$p(\mathbf{x}, \Theta) = \sum_{i=1}^{M} \alpha_i p_i(\mathbf{x}|\mu_i, \Sigma_i) \tag{3}$$

$$p_i(\mathbf{x}|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu_i)^T \Sigma_i^{-1}(\mathbf{x}-\mu_i)} \tag{4}$$

given sample $X = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ (training data). In the above formulas p_i is the density of the Gaussian distribution with mean μ_i and covariance matrix Σ_i. α_i is the weight of the component i of the mixture, M is the number of mixture components or clusters and is fixed and given, and d is the dimensionality of the space.

The EM algorithm for estimating the parameters of the Gaussian components proceeds by repeatingly applying the following two steps until values of the estimates do not change significantly:

Expectation (E step):

$$h_{ji} = \frac{\alpha_i p_i(\mathbf{x_j}|\mu_i, \Sigma_i)}{\sum_{k=1}^{M} \alpha_k p_k(\mathbf{x_j}|\mu_k, \Sigma_k)} \tag{5}$$

Maximization (M step):

$$\alpha_i = \frac{1}{N}\sum_{j=1}^{N} h_{ij}, \quad \mu_i = \frac{\sum_{j=1}^{N} h_{ij}\mathbf{x}_j}{\sum_{j=1}^{N} h_{ij}}$$

$$\Sigma_i = \frac{\sum_{j=1}^{N} h_{ij}(\mathbf{x}_j - \mu_i)(\mathbf{x}_j - \mu_i)^T}{\sum_{j=1}^{N} h_{ij}}$$

3. PREVIOUS SOLUTIONS TO LINEAR REGRESSION TREE CONSTRUCTION

In this section we analyze some of the previously proposed construction algorithms for linear regression trees and, for each, we point major drawbacks.

3.1 Quinlan's construction algorithm

For efficiency reasons, the algorithm proposed by Quinlan [16] *pretends* that a regression tree with constant models in the leaves is constructed until the tree is fully grown, when linear models are fit on the datapoints available at leaves. This is equivalent to using the split selection criterion in Equation 2 during the growing phase. Then linear regressors in the leaves are constructed by performing another pass over the data in which the set of datapoints from the training examples corresponding to each of the leaves is determined and the least square linear problem for these datapoints is formed and solved (using the SVD decomposition [10]).

The same approach was latter used by Torgo [18, 19] with more complicated models in the leaves like kernels and local polynomials.

As pointed out by Karalic [11] the variance of the output variable is a poor estimator of the purity of the fit when linear regressors are used in the leaves since the points can be arranged along a line (so the error of the linear fit is almost zero) but they occupy a significant region (so the variance is large). To correct this problem, he suggested that the following impurity function should be used:

$$\mathrm{Err}_l(T) \overset{\mathrm{def}}{=} E\left[(Y - E\left[f(\mathbf{X})|T\right])^2\right] \tag{6}$$

where $f(\mathbf{x}) = [1\ \mathbf{x}^T]\mathbf{c}$ is the linear regressor with the smallest least square error. It is easy to see (see for example [10]) that

$\mathbf{c}$ is the solution of the LSE equation:

$$E\left[\begin{bmatrix}1 & \mathbf{X}^T \\ \mathbf{X} & \mathbf{X}\mathbf{X}^T\end{bmatrix}\middle| T\right]\mathbf{c} = E\left[\begin{bmatrix}1 \\ \mathbf{X}\end{bmatrix} Y \middle| T\right] \tag{7}$$

To see more clearly that $\mathrm{Err}(T)$ given by Equation 2 is not appropriate for the linear regressor case, consider the situation in Figure 2. The two thick lines represent a large

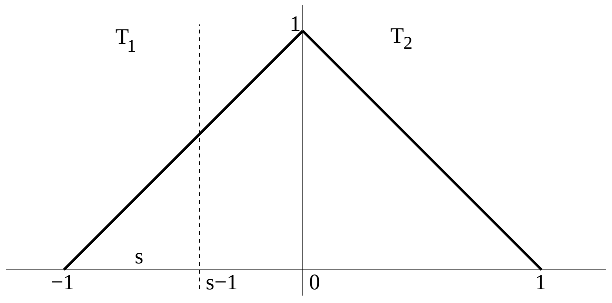

Figure 2: Example of situation where average based decision is different from linear regression based decision

number of points (possibly infinite). The best split for the linear regressor is $x = 0$ and the fit is perfect after the split (thus $\mathrm{Err}_l(T_1) = \mathrm{Err}_l(T_2) = 0$). Obviously the split has the maximum possible gini gain (1/12).

For the case when $\mathrm{Err}(T)$ is used, $E[Y|T] = 1/2$ so $\mathrm{Err}(T) = 1/12$. To determine the split point for this situation suppose the split point is $s-1$ in Figure 2. The points with property $x < s-1$ belong to T_1 and the rest to T_2. Then $E[Y|T_1] = s/2$, $\mathrm{Err}(T_1) = s^3/12$, $E[Y|T_2] = (-2+s^2)/2(-2+s)$ and $\mathrm{Err}(T_2) = (4-8s+12s^2-8s^3+s^4)/(24-12s)$. Thus by doing the split the impurity decreases by $\Delta\mathrm{Err}(T) = (1-s)^2 s/4(2-s)$ The extremum points in the interval $[0,1]$ are $s = 1$ and $s = (3-\sqrt{5})/2$. Looking at the second derivative in these points one can observe that $\Delta\mathrm{Err}(T)$ has a minimum in $s = 1$ and a maximum in $s = (3-\sqrt{5})/2$. Thus the maximum impurity decrease is obtained if the split point is $-(\sqrt{5}-1)/2 = -0.618034$ or symmetrically 0.618034. Either of this splits is very far from the split obtained using $\mathrm{Err}_l(T)$ (at point 0), thus splitting the points in proportion 19% to 81% instead of the ideal 50% to 50%.

This example suggests that the split point selection based on $\mathrm{Err}(T)$ produces an unnecessary fragmentation of the data that is not related to the natural organization of the datapoints for the case of linear regression trees. This fragmentation produces unnecessarily large and unnatural trees, anomalies that are not corrected by the pruning stage. Indeed, when we used a dataset with the triangular shape described above as the input to a regression tree construction algorithm that used $\mathrm{Err}(T)$ from Equation 2 as split criterum we obtained the following split points starting from the root and navigating breadth first for three levels: 0.6185, -0.5255, 0.8095, -0.7625, 0.3585, 0.7145, 0.9055. Splits are not only anintuitive but the generated tree is very unbalanced. Note that this example is not an extreme case but rather a normal one so this behavior is probably the norm not the exception.

3.2 Karalic's construction algorithm

Using the split criterion in Equation 6 the problem mentioned above is avoided and much higher quality trees are built. If exhaustive search is used to determine the split point, the computational cost of the algorithm becomes prohibitively expensive for large datasets for two main reasons:

- If the split attribute is continuous, all possible values of this attribute have to be considered as split points. For each of them a linear system has to be formed and solved. Even if the matrix and the vector that form the linear system are maintained incrementally (which can be dangerous from numerical stability point of view), for every level of the tree constructed, a number of linear systems equal to the size of the dataset have to be solved.
- If the split attribute is discrete the situation is worse since Theorem 9.4 in [4] does not seem to apply for this split criterion. This means that an exponential, in the size of the domain of the split variable, number of linear systems have to be formed and solved.

The first problem can be alleviated if a sample of the points available are considered as split points. Even if this simplification is made, the datapoints have to be sorted in every intermediate node on all the possible split attributes. Also, it is not clear how this modifications influence the accuracy of the generated regression trees. The second problem seems unavoidable if exhaustive search is used.

3.3 Chaudhuri's et al. construction algorithm

In order to avoid forming and solving so many linear systems, Chaudhuri et al. [5] proposed to locally classify the datapoints available at an intermediate node based on the sign of the residual with respect to the least square error linear model. For the datapoints in Figure 3 (the set of datapoints is identical to the one in Figure 2) this corresponds to points above and below the dashed line. As it can be observed, when projected on the X axis, the negative class surrounds the positive class so two split points are necessary to differentiate between them (the node has to be split into three parts). When the number of predictor attributes is greater than 1 (multidimensional case), the separating surface between class labels $+$ and $-$ is nonlinear. Moreover, if best regressors are fit in these two classes, the prediction is only slightly improved. The solution adopted by Chaudhuri et al. is to use Quadratic Discriminant Analysis (QDA) to determine the split point. This usually leads to choosing as split point approximatively the mean of the dataset, irrespective of where the optimal split is, so the reduction is not very useful. For this reason GUIDE [13] uses this method to select the split attribute but not the split point.

4. SCALABLE LINEAR REGRESSION TREES

For constant regression trees, algorithms for scalable classification trees can be straightforwardly adapted [9]. The main obstacle in doing the same thing for linear regression trees is the observation previously made that the problem of partitioning the domain of a discrete variable in two parts is intractable. Also the amount of sufficient statistics that has to be maintained goes from two real numbers for constant regressors (mean and mean of square) to quadratic in the number of regression attributes (to maintain the matrix

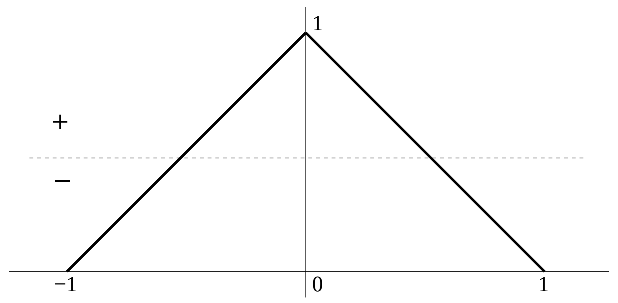

Figure 3: Example where classification on sign of residuals is unintuitive.

Input: node T, data-partition D
Output: regression tree $\mathcal{T}$ for D rooted at T

Linear regression tree construction algorithm:

```
BuildTree(node $T$, data-partition $D$)
(1)  normalize datapoints to unitary sphere
(2)  find two Gaussian clusters in regressor–output space (EM)
(3)  label datapoints based on closeness to these clusters
(4)  foreach split attribute
(5)    find best split point and determine its gini gain
(6)  endforeach
(7)  let $X$ be the attribute with the greatest gini gain and
         $Q$ the coresponding best split predicate set
(8)  if ($T$ splits)
(9)    partition $D$ into $D_1, D_2$ based on $Q$ and label node $T$
           with split attribute $X$
(10)   create children nodes $T_1, T_2$ of $T$ and label
           the edge $(T, T_i)$ with predicate $q_{(T,T_i)}$
(11)   BuildTree($T_1$, $D_1$); BuildTree($T_2$, $D_2$)
(12) else
(13)   label $T$ with the least square linear regressor of $D$
(14) endif
```

Figure 4: SECRET algorithm

$A^T A$ that defines the linear system). This can be a problem also.

In this work we make the distinction in [13] between predictor attributes: (1) *discrete attributes* – used only in the split predicates in intermediate nodes in the regression tree, (2) *split continuous attributes* – continuous attributes used only for splitting, (3) *regression attributes* – continuous attributes used in the linear combination that specifies the linear regressors in the leaves as well as for specifying split predicates. By allowing some continuous attributes to participate in splits but not in regression in the leaves we add greater flexibility to the learning algorithm. The partitioning of the continuous attributes in split and regression is beyond the scope of the paper (and is usually performed by the user [13]).

The main idea behind our algorithm is to locally transform the regression problem into a classification problem by first identifying two general Gaussian distributions in the regressor attributes–output space using the EM algorithm for Gaussian mixtures and then classifying the datapoints based on the probability of belonging to these two distributions. Classification tree techniques are then used to select the split attribute and the split point. Our algorithm, called SECRET, is summarized in Figure 4.

The role of EM is to find two natural classes in the data that have approximatively a linear organization. The role of

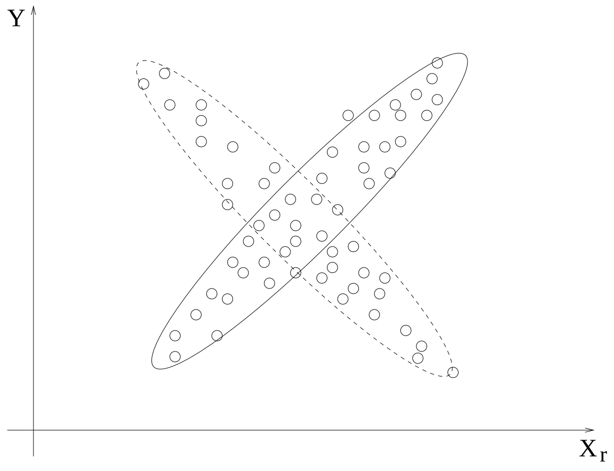

Figure 5: Projection on X_r, Y space of training data.

the classification is to identify predictor attributes that can make the difference between these two classes in the input space. To see this more clearly suppose we are in the process of building a linear regression tree and we have to decide on the split attribute and split point for the node T. Suppose the set of training examples available at node T contains tuples with three components: a regressor attribute X_r, a discrete attribute X_d and the predicted attribute Y. The projection of the training data on the X_r, Y space might look like Figure 5. The datapoints are approximatively organized in two clusters with Gaussian distributions that are marked as ellipsoids. Differentiating between the two clusters is crucial for prediction, but information in the regression attribute is not sufficient to make this distinction even though within a cluster they can do good predictions. The information in the discrete attribute X_d can make this distinction, as can be observed from Figure 6 where the projection is made on the X_d, X_r, Y space. If more split attributes had been present, a split on X_d would have been preferred since the resulting splits are pure.

Observe that the use of the EM algorithm for Gaussian mixtures is very limited since we have only two mixtures and thus the likelihood function has a simpler form which means less local maxima. Since EM is sensitive to distances, before running the algorithm, training data has to be normalized by performing a linear transformation that makes the data look as close as possible to a unitary sphere with the center in the origin. Experimentally we observed that, with this transformation and in this restricted scenario, the EM algorithm with clusters initialized randomly works well.

We describe first how the EM algorithm can be implemented efficiently followed by details on the integration of the resulting mixtures with the split selection procedure and the linear regression in the leaves.

4.1 Efficient Implementation of EM Algorithm

The following two ideas are used to implement efficiently the EM algorithm: (1) steps E and M are performed simultaneously, which means that quantities h_{ij} do not have to be stored explicitly, (2) all the operations are expressed in terms of Cholesky decomposition G_i of covariance matrix $\Sigma_i = G_i G_i^T$. G_i has the useful property that is lower di-

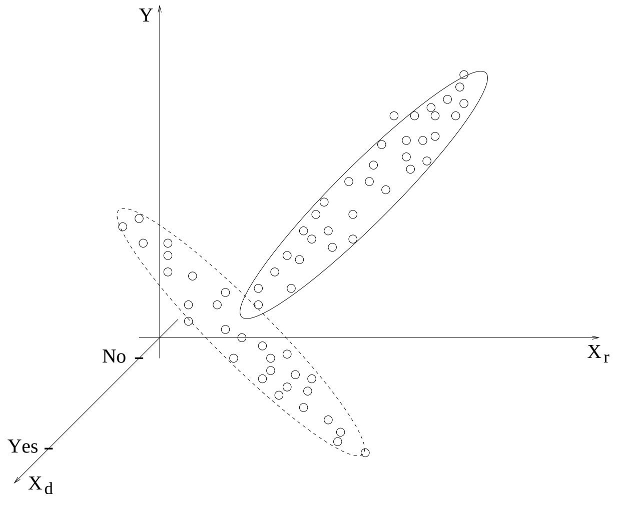

Figure 6: Projection on X_d, X_r, Y space of same training data as in Figure 5

agonal, so solving linear systems takes quadratic effort in the number of dimensions and computing the determinant is linear in the number of dimensions. Note that this modifications can be made in addition to the techniques used in [3] to make the EM algorithm scalable.

Using the Cholesky decomposition we immediately have $\Sigma_i^{-1} = G_i^{-1T} G_i^{-1}$ and $|\Sigma_i| = |G_i|^2$. Substituting in Equation 4 we get:

$$p_i(\mathbf{x}|\mu_i, G_i) = \frac{1}{(2\pi)^{d/2}|G_i|} e^{-\frac{1}{2}\|G_i^{-1}(\mathbf{x}-\mu_i)\|^2}$$

Quantity $\mathbf{x}' = G_i^{-1}(\mathbf{x} - \mu_i)$ can be computed by solving the linear system $G_i\mathbf{x}' = \mathbf{x} - \mu_i$ and takes quadratic effort. For this reason the inverse of G_i needs not be precomputed since solving the linear system takes as much time as vector matrix multiplication.

The following quantities have to be maintained incrementally for each cluster:

$$s_i = \sum_{j=1}^{N} h_{ij}, \quad s_{\mathbf{x},i} = \sum_{j=1}^{N} h_{ij}\mathbf{x}_j, \quad S_i = \sum_{j=1}^{N} h_{ij}\mathbf{x}_j^T\mathbf{x}_j$$

and for every training example $\mathbf{x}_j$ quantities h_{ij} are computed with the formula in Equation 5. and are discarded after updating $s_i, s_{\mathbf{x},i}, S_i$ for every cluster i.

After all the training examples have been seen, the new parameters of the two distributions are computed with the formulas:

$$\alpha_i = \frac{s_i}{N}, \quad \mu_i = \frac{s_{\mathbf{x},i}}{s_i},$$
$$\Sigma_i = \frac{S_i}{s_i} - {\mu_\mathbf{i}}^T\mu_\mathbf{i}, \quad G_i = \mathrm{Chol}(\Sigma_i)$$

Moreover, if the datapoints are coming from a Gaussian distribution with mean μ_i and covariance matrix $G_iG_i^T$ the transformation $\mathbf{x}' = G_i^{-1}(\mathbf{x} - \mu_i)$ results in datapoints with Gaussian distribution with mean 0 and identity covariance matrix. This means that this transformation can be used for data normalization in the tree growing phase.

4.2 Split point and attribute selection

Once the two Gaussian mixtures are identified, the datapoints can be labeled based on the closeness to the two clusters (i.e. if a datapoint is closer to cluster 1 than cluster 2 it is labeled with class label 1, otherwise it is labeled with class label 2). In this moment, locally, split point and attribute selection methods from classification tree construction can be used.

We are using gini gain as the split selection criteria to find the split point. That is, for each attribute (or collection of attributes for oblique splits) we determine the best split point and compute the gini gain. Then the predictor attribute with the greatest gini gain is chosen as split attribute.

For the discrete attributes the algorithm of Breiman et al. [4] finds the split point in time linear in the size of the domain of the discrete attribute (since we only have two class labels). We use this algorithm, unchanged, in the present work to find the split point for discrete attributes.

4.2.1 Split point selection for continous attributes

Since the EM algorithm for Gaussian mixtures produces two normal distributions, it is reasonable to assume that the projection of the datapoints with the same class label on a continuous attribute X has also a normal distribution. The split point that best separates the two normal distributions can be found using Quadratic Discriminant Analysis (QDA). The reason for preferring QDA to a direct minimization of the gini gain is the fact that it gives qualitatively similar splits but requires less computational effort [14]. Let $\alpha_i, \eta_i, \sigma_i^2$ be the apriory probability, mean and variance of the distribution $i \in 1, 2$. The solution of the QDA problem is the point between the centers of the two normal distributions where the two densities are equal. Thus the separation point η satisfies:

$$\alpha_1 \frac{1}{\sigma_1\sqrt{2\pi}} e^{-(\eta_1-\eta)^2/2\sigma_1^2} = \alpha_2 \frac{1}{\sigma_2\sqrt{2\pi}} e^{-(\eta_2-\eta)^2/2\sigma_2^2}$$

that is equivalent to the second order equation:

$$\eta^2\left(\frac{1}{\sigma_1^2} - \frac{1}{\sigma_2^2}\right) - 2\eta\left(\frac{\eta_1}{\sigma_1^2} - \frac{\eta_2}{\sigma_2^2}\right) + \frac{\eta_1^2}{\sigma_1^2} - \frac{\eta_2^2}{\sigma_2^2} = 2\ln\frac{\alpha_1}{\alpha_2} - \ln\frac{\sigma_1^2}{\sigma_2^2}$$

From the two solutions of the equation the one between η_1 and η_2 is preferred. If σ_1^2 is very close to σ_2^2, solving the second order equation is not numerically stable. In this case it is preferable to solve the linear equation:

$$2\eta(\eta_1 - \eta_2) = \eta_1^2 - \eta_2^2 - 2\sigma_1^2 \ln\frac{\alpha_1}{\alpha_2}$$

From Equation 1 it is obvious that to compute the gini gain all we need to compute is:$P[x \in C_1|x \le \eta]$ and $P[x \in C_2|x \le \eta]$, and put $P[x \in C_1] = \alpha_1$ and $P[x \le \eta] = P[x \in C_1]P[x \in C_1|x \le \eta] + P[x \in C_2]P[x \in C_2|x \le \eta]$. $P[x \in C_1|x \le \eta]$ is the p-value of the normal distribution with mean η_1 and variance σ_1^2 with respect to $x \le \eta$. That is:

$$\begin{aligned} P[x \in C_1|x \le \eta] &= \int_{x\le\eta} \frac{1}{\sigma_1\sqrt{2\pi}} e^{-(x-\eta_1)^2/2\sigma_1^2} dx \\ &= \frac{1}{2}\left(1 + \mathrm{Erf}\left(\frac{\eta_1 - \eta}{\sigma_1\sqrt{2}}\right)\right) \end{aligned}$$

$P[x \in C_2|x \le \eta]$ is obtained similarly.

4.2.2 *Finding a good oblique split for two Gaussian mixtures*

Ideally, given two Gaussian distributions, we would like to find the separating hyperplane that maximizes the gini gain. Fukanaga showed that the problem of minimizing the expected value of the 0-1 loss (the classification error function) generates an equation involving the normal of the hyperplane that is not solvable algebraically [8]. Following the same treatment, it is easy to see that the problem of minimizing the gini gain generates the same equation. A good solution to the problem of determining a separating hyperplane can be found using Linear Discriminant Analysis (LDA) [8].

The solution of an LDA problem for two mixtures consists of minimizing Fisher's separability criterion:

$$J(\mathbf{n}) = \frac{\mathbf{n}^T \Sigma_w \mathbf{n}}{\mathbf{n}^T \Sigma_b \mathbf{n}}$$

with

$$\Sigma_w = \sum_{i=1,2} \alpha_i (\mu - \mu_i)(\mu - \mu_i)^T, \quad \mu = \sum_{i=1,2} \alpha_i \mu_i$$

$$\Sigma_b = \sum_{i=1,2} \alpha_i \Sigma_i$$

which has as result a vector $\mathbf{n}$, with the property that the projections on this vector of the two Gaussian distributions is as separated as possible. The solution of the optimization problem is [8]:

$$\mathbf{n} = \frac{\Sigma_w^{-1}(\mu_1 - \mu_2)}{\|\Sigma_w^{-1}(\mu_1 - \mu_2)\|^2}$$

The value of Fisher's criterion is invariant to the choice of origin on the projection so we can make the projection on the line given by the vector $\mathbf{n}$, that optimizes Fisher's criterion, and the origin of the coordinate system.

The two multidimensional Gaussian distributions are transformed into unidimensional normal distributions on the projection line with means $\eta_i = \mathbf{n}^T \mu_i$ and the variances $\sigma_i^2 = \mathbf{n}^T \Sigma_i \mathbf{n}$ for $i = 1, 2$, the coordinates being line coordinates with the coordinate of the projection of the origin as the 0.

In this moment the QDA, as in the previous section, can be used to find the split point η on the projection. The point η on the projection line corresponds to $\eta\mathbf{n}$ in the initial space. The equation of the separating hyperplane that has $\mathbf{n}$ as normal and contains point $\eta\mathbf{n}$ is $\mathbf{n}^T(\mathbf{x} - \eta\mathbf{n}) = 0 \Leftrightarrow \mathbf{n}^T\mathbf{x} - \eta = 0$. With this, a point $\mathbf{x}$ belongs to the first partition if $\text{sign}(\eta_1 - \eta)(\mathbf{n}^T\mathbf{x} - \eta) \geq 0$. The hyperplane that contains this point of the projection line and that is perpendicular to the projection line is a good separator of the two multidimensional Gaussian distributions.

In order to be able to compare the efficacy of the split with other splits, we have to compute its gini gain. The same method as for the case of unidimensional splits of continuous variables can be used here. The only unsolved problem is computing the p-value of a Gaussian distribution with respect to a half-space. The solution is given by the following result:

PROPOSITION 1. *For a Gaussian distribution with mean μ and covariance matrix $\Sigma = GG^T$, positive definite, and density $p_{\mu,\Sigma}(\mathbf{x})$ and a hyperplane with normal $\mathbf{n}$ that contains the point $\mathbf{x}_c$, the p-value with respect to the hyperplane is:*

$$P[\mathbf{n}^T(\mathbf{x} - \mathbf{x}_c) \geq 0 | \mu, \Sigma] = \int_{\mathbf{n}^T(\mathbf{x}-\mathbf{x}_c)\geq 0} p_{\mu,\Sigma}(\mathbf{x}) d\mathbf{x}$$
$$= \frac{\sigma}{2\sqrt{|\Sigma||S|}} \left(1 + \text{Erf}\left(\frac{\mu_1'}{\sigma\sqrt{2}}\right)\right)$$

where

$$\Sigma'^{-1} = \begin{pmatrix} s & \mathbf{w}^T \\ \mathbf{w} & S \end{pmatrix} = M^T \Sigma^{-1} M$$

with M orthogonal such that $M^T\mathbf{n} = \mathbf{e}_1$, $\sigma = 1/\sqrt{s - \mathbf{w}^T S^{-1}\mathbf{w}}$ and $\mu' = M^T(\mu - \mathbf{x}_c)$.

PROOF. See Appendix A. □

4.2.3 *Finding linear regressors*

If the current node is a leaf or in preparation for the situation that all the descendents of this node are pruned we have to find the best linear regressor that fits the training data. We identified two ways the LSE linear regressor can be computed.

The first method consist of a traversal of the original dataset and an identification of the subset that falls into this node. The least square linear system in Equation 7 is formed with these datapoints and solved. Note that, in the case that all the sufficient statistics can be maintained in main memory, a single traversal of the training dataset per tree level will suffice.

The second method uses the fact that the split selection method tries to find a split attribute and a split point that can differentiate best between the two Gaussian mixtures found in the regressor–output space. The least square problem can be solved at the level of each of these mixtures under the assumption that the distribution of the datapoints is normal with the parameters identified by the EM algorithm. This method is less precise since the split is usually not perfect but can be used when the number of traversals over the dataset becomes a concern.

5. EMPIRICAL EVALUATION

In this section we present the results of an extensive experimental study of SECRET, the linear regression tree construction algorithm we propose. The purpose of the study was twofold: (1) to compare the accuracy of SECRET with GUIDE [13], a state-of-the-art linear regression tree algorithm and (2) to compare the scalability properties of SECRET and GUIDE through running time analysis.

The main findings of our study are:

- **Accuracy of prediction.** SECRET is more accurate than GUIDE on three datasets, as accurate on six datasets and less accurate on three datasets. This suggests that overall the prediction accuracy to be expected from SECRET is comparable to the accuracy of GUIDE. On four of the datasets, the use of oblique splits resulted in significant improvement in accuracy.

- **Scalability to large datasets.** For datasets of small to moderate sizes (up to 5000 tuples), GUIDE slightly outperforms SECRET. The behavior for large datasets of the two methods is very different; for datasets with 256000 tuples and 3 attributes, SECRET runs about 200 times faster than GUIDE. Even if GUIDE considers only 1% of the points available as possible split points, SECRET still runs 20 times faster. Also, there is no significant change in running time when SECRET produces oblique splits.

5.1 Experimental testbed and methodology

GUIDE [13] is a regression tree construction algorithm that was designed to be both accurate and fast. The extensive study by Loh [13] showed that GUIDE outperforms previous regression tree construction algorithms and compares favorably with MARS [7], a state-of-the-art regression algorithm based on spline functions. GUIDE uses statistical techniques to pick the split variable and can use exhaustive search or just a sample of the points to find the split point. In our accuracy experiments we set up GUIDE to use exhaustive search. For the scalability experiments we report running times for both the exhaustive search and split point candidate sampling of size 1%.

For the experimental study we used nine real life and three synthetic datasets. All datasets except 3DSin have been used before extensively.

Real life datasets:

Abalone Dataset from UCI machine learning repository used to predict the age of abalone from physical measurements. Contains 4177 cases with 8 attributes (1 nominal and 7 continuous).

Baseball Dataset from UCI repository, containing information about baseball players used to predict their salaries. Consists of 261 cases with 20 attributes (3 nominal and 17 continuous).

Boston Data containing characteristics and prices of houses around Boston, from UCI repository. Contains 506 cases with 14 attributes (2 nominal and 12 continuous).

Kin8nm Data containing information on the forward kinematics of an 8 link robot arm from the DVELVE repository. Contains 8192 cases with 8 continuous attributes.

Mpg Subset of the auto-mpg data in the UCI repository (tuples with missing values were removed). The data contains characteristics of automobiles that can be used to predict gas consumption. Contains 392 cases with 8 attributes (3 nominal and 5 continuous).

Mumps Data from `SatLib` archive containing incidence of mumps in each of the 48 contiguous states from 1953 to 1989. The predictor variables are year and longitude and latitude of state center. The dependent variable is the logarithm of the number of mumps cases. Contains 1523 cases with 4 continuous attributes.

Stock Data containing daily stock of 10 aerospace companies from `SatLib` repository. The goal is to predict the stock of the 10th company from the stock of the other 9. Contains 950 cases with 10 continuous attributes.

TA Data from UCI repository containing information about teaching assistants at University of Wisconsin. The goal is to predict their performance. Contains 151 cases with 6 attributes (4 nominal and 2 continuous).

Tecator Data from `SatLib` archive containing characteristics of spectra of pork meat with the purpose of predicting the fat content. We used the first 10 principal components of the wavelengths to predict the fat content. Contains 240 cases with 11 continuous attributes.

Synthetic datasets:

Cart Synthetic dataset proposed by Breiman et al.([4] p.238) with 10 predictor attributes: $X_1 \in \{-1,1\}$, $X_i \in \{-1,0,1\}, i \in \{2\ldots10\}$ and the predicted attribute determined by if $X_1 = 1$ then $Y = 3 + 3X_2 + 2X_3 + X_4 + \sigma(0,2)$ else $Y = -3 + 3X_5 + 2X_6 + X_7 + \sigma(0,2)$. We interpreted all the 10 predictor attributes as discrete attributes.

Fried Artificial dataset used by Friedman [7] containing 10 continuous predictor attributes with independent values uniformly distributed in the interval $[0,1]$. The value of the predictor variable is obtained with the equation: $Y = 10\sin(\pi X_1 X_2) + 20(X_3 - 0.5)^2 + 10X_4 + 5X_5 + \sigma(0,1)$.

3DSin Artificial dataset containing 2 continuous predictor attributes uniformly distributed in interval $[-3,3]$, with the output defined as $Y = 3\sin(X_1)\sin(X_2)$. There is no noise added.

We performed all the experiments reported in this paper on a Pentium III 933MHz running Redhat Linux 7.2.

5.2 Experimental results: Accuracy

For each experiment with real datasets we used a random partitioning into 50% of datapoints for training, 30% for pruning and 20% for testing. For the synthetic datasets we generated randomly 16384 tuples for training, 16384 tuples for pruning and 16384 tuples for testing for each experiment. We repeated each experiment 100 times in order to get accurate estimates. For comparison purposes we built regression trees with both constant (by using all the continuous attributes as split attributes) and linear (by using all continuous attributes as regressor attributes) regression models in the leaves. In all the experiments we used Quinlan's resubstitution error pruning [17]. For both algorithms we set the minimum number of datapoints in a node to be considered for splitting to 1% of the size of the dataset, which resulted in trees at the end of the growth phase with around 75 nodes.

Table 1 contains the average mean square error and its standard deviation for GUIDE, SECRET and SECRET with oblique splits (SECRET(O)) with constant (left part) and linear (right part) regressors in the leaves, on each of the twelve datasets. GUIDE and SECRET with linear regressor in the leaves have equal accuracy (we considered accuracies equal if they were less than three standard deviations away from each other) on six datasets (Abalone, Boston, Mpg, Stock, TA and Tecator), GUIDE wins on three datasets (Baseball, Mumps and Fried) and SECRET wins on the remaining three (Kin8nm, 3DSin and Cart). These findings suggest that the two algorithms are comparable from the accuracy point of view, neither dominating the other. The use of oblique splits in SECRET made a big difference in four datasets (Kin8nm 27%, Stock 24%, Tecator 35% and 3DSin 45%). These datasets usually have less noise and are complicated but smooth (so they offer more opportunities for *intelligent* splits). At the same time the use of oblique splits resulted in significantly worse performance on two of the datasets (Baseball 13% and Fried 19%).

5.3 Experimental results: Scalability

We chose to use only synthetic datasets for scalability experiments since the sizes of the real datasets are too small.

	Constant Regressors			Linear Regressors		
	GUIDE	SECRET	SECRET(O)	GUIDE	SECRET	SECRET(O)
Abalone	5.32±0.05	5.50±0.10	5.41±0.10	*4.63±0.04*	4.67±0.04	4.76±0.05
Baseball	0.224±0.009	0.200±0.008	0.289±0.012	**0.173±0.005**	0.243±0.011	0.280±0.009
Boston	**23.34±0.72**	28.00±0.92	30.91±0.94	40.63±6.63	24.01±0.69	26.11±0.66
Kin8nm	0.0419±0.0002	0.0437±0.0002	0.0301±0.0003	0.0235±0.0002	0.0222±0.0002	**0.0162±0.0001**
Mpg	**12.94±0.33**	30.09±2.28	26.26±2.45	34.92±21.92	15.88±0.68	16.76±0.74
Mumps	1.34±0.02	1.59±0.02	1.56±0.02	**1.02±0.02**	1.23±0.02	1.32±0.04
Stock	2.23±0.06	2.20±0.06	2.18±0.07	1.49±0.09	1.35±0.05	**1.03±0.03**
TA	0.74±0.02	0.69±0.01	*0.69±0.01*	0.81±0.04	0.72±0.01	0.79±0.08
Tecator	57.59±2.40	49.72±1.72	28.21±1.75	13.46±0.72	12.08±0.53	**7.80±0.53**
3DSin	0.1435±0.0020	0.4110±0.0006	0.2864±0.0077	0.0448±0.0018	0.0384±0.0026	**0.0209±0.0004**
Cart	1.506±0.005	**1.171±0.001**	N/A	N/A	N/A	N/A
Fried	7.29±0.01	7.45±0.01	6.43±0.03	**1.21±0.00**	1.26±0.01	1.50±0.01

Table 1: Accuracy on real (upper part) and synthetic (lower part) datasets of GUIDE and SECRET. In parenthesis we indicate O for orthogonal splits. The winner is in bold face if it is statistically significant and in italics otherwise.

Size	GUIDE	GUIDE(S)	SECRET	SECRET(O)
250	0.07	0.05	0.21	0.21
500	0.13	0.07	0.33	0.34
1000	0.30	0.12	0.55	0.58
2000	0.94	0.24	1.08	1.12
4000	3.28	0.66	2.11	2.07
8000	12.58	2.40	4.07	4.12
16000	48.93	9.48	8.16	8.37
32000	264.50	43.25	16.71	16.19
64000	1389.88	184.50	35.62	35.91
128000	6369.94	708.73	73.35	71.67
256000	25224.02	2637.94	129.95	131.70

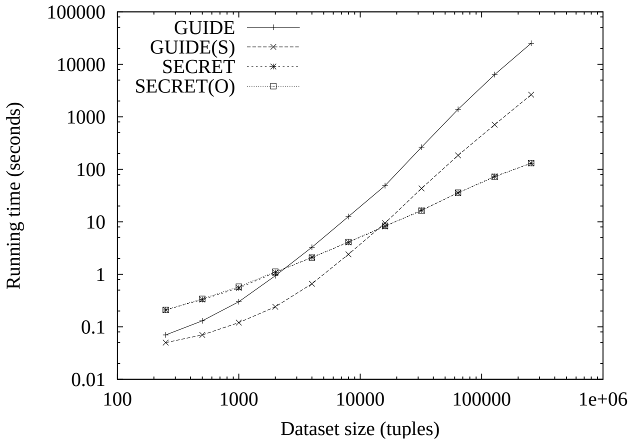

Figure 7: Running time (in seconds) of GUIDE, GUIDE with 0.01 of point as split points, SECRET and SECRET with oblique splits for synthetic dataset 3DSin (3 continuous attributes).

Size	GUIDE	GUIDE(S)	SECRET	SECRET(O)
250	0.09	0.07	0.47	0.43
500	0.17	0.14	0.87	0.92
1000	0.36	0.28	1.85	1.83
2000	1.12	0.80	3.58	3.69
4000	2.90	2.38	7.33	7.36
8000	10.46	8.43	13.77	14.05
16000	42.16	33.09	27.80	28.68
32000	194.63	123.63	56.87	58.01
64000	1082.70	533.16	122.26	124.60
128000	4464.88	1937.94	223.42	222.75
256000	18052.16	8434.33	460.12	470.68

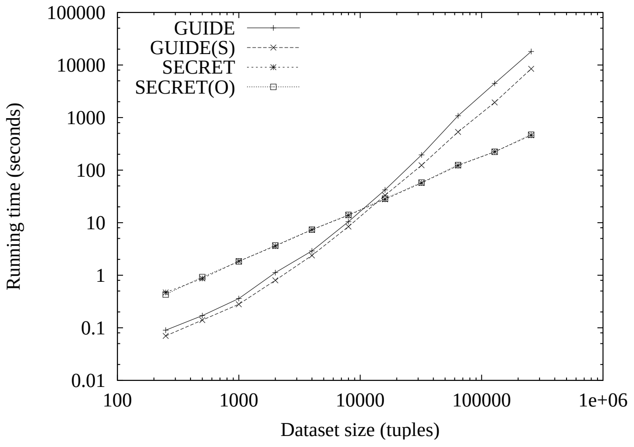

Figure 8: Running time (in seconds) of GUIDE, GUIDE with 0.01 of point as split points, SECRET and SECRET with oblique splits for synthetic dataset Fried (11 continuous attributes).

The learning time of both GUIDE and SECRET is mostly dependent on the size of the training set and on the number of attributes, as is confirmed by some other experiments we are not reporting here. As in the case of accuracy experiments, we set the minimum number of datapoints in a node to be considered for further splits to 1% of the size of the training set. We measured only the time to grow the trees, ignoring the time necessary for pruning and testing. The reason for this is the fact that pruning and testing can be implemented efficiently and for large datasets do not make a significant contribution to the running time. For GUIDE we report running times for both exhaustive search and sample split point (only 1% of the points available in a node are considered as possible split points), denoted by GUIDE(S).

Results of experiments with the 3DSin dataset and Fried dataset are depicted in Figures 7 and 8 respectively. A number of observations are apparent from these two sets of results: (1) the performance of the two versions of SECRET (with and without oblique splits) is virtually indistinguishable, (2) the running time of both versions of GUIDE is quadratic in size for large datasets, (3) as the number of attributes went up from 3 (3DSin) to 11 (Fried) the computation time for GUIDE(S), SECRET and SECRET(O) went up about 3.5 times but went slightly down for GUIDE, (4) for large datasets (256000 tuples) SECRET is two orders of magnitude faster than GUIDE and one order of magnitude faster than GUIDE(S). It is also worth pointing out that, for SECRET, most of the time is spent in the EM algorithm. If used, sampling would not decrease the precision of EM much and at the same time would considerably decrease the computation time. For this reason the comparison with GUIDE(S) is not fair, nevertheless starting from medium sized datasets SECRET outperforms significantly the sampled version of GUIDE.

6. CONCLUSIONS

In this paper we introduced SECRET, a new linear regression tree construction algorithm designed to overcome the scalability problems of previous algorithms. The main idea is, for every intermediate node, to find two Gaussian clusters in the regressor–output space and then to classify the datapoints based on the closeness to these clusters. Techniques from classification tree construction are then used locally to choose the split variable and split point. In this way the problem of forming and solving a large number of linear systems, required by an exhaustive search algorithm, is avoided entirely. Moreover, this reduction to a local classification problem allows us to efficiently build regression trees with oblique splits. Experiments on real and synthetic datasets showed that the proposed algorithm is as accurate as GUIDE, a state-of-the-art regression tree algorithm if normal splits are used, and on some datasets up to 45% more accurate if oblique splits are used. At the same time our algorithm requires significantly smaller computational resources for large datasets.

7. ACKNOWLEDGEMENTS

We would like to thank Wei-Yin Loh for providing help with GUIDE, Rich Caruana for making comments on an early draft of this paper and Rimon Barr for suggesting the name SECRET for our algorithm.

8. REFERENCES

[1] W. P. Alexander and S. D. Grimshaw. Treed regression. *Journal of Computational and Graphical Statistics*, (5):156–175, 1996.

[2] J. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical report, University of California at Berkeley, 1997.

[3] P. S. Bradley, U. M. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Knowledge Discovery and Data Mining*, pages 9–15, 1998.

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

[5] P. Chaudhuri, M.-C. Huang, W.-Y. Loh, and R. Yao. Piecewise-polynomial regression trees. *Statistica Sinica*, 4:143–167, 1994.

[6] N. M. Dempster,A.P. Laird and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. R. Statist. Soc. B*, 39:185–197, 1977.

[7] J. H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19:1–141 (with discussion), 1991.

[8] K. Fukanaga. *Introduction to Statistical Pattern Recognition, Second edition.* Academic Press, 1990.

[9] J. Gehrke, R. Ramakrishnan, and V. Ganti. Rainforest – a framework for fast decision tree construction of large datasets. In *Proceedings of the 24th International Conference on Very Large Databases*, pages 416–427. Morgan Kaufmann, August 1998.

[10] G. H. Golub and C. F. V. Loan. *Matrix Computations.* Johns Hopkins, 1996.

[11] A. Karalic. Linear regression in regression tree leaves. In *International School for Synthesis of Expert Knowledge, Bled,Slovenia*, 1992.

[12] K.-C. Li, H.-H. Lue, and C.-H. Chen. Interactive tree-structured regression via principal hessian directions. *journal of the American Statistical Association*, (95):547–560, 2000.

[13] W.-Y. Loh. Regression trees with unbiased variable selection and interaction detection. *Statistica Sinica*, 2002. in press.

[14] W.-Y. Loh and Y.-S. Shih. Split selection methods for classification trees. *Statistica Sinica*, 7(4), 1997.

[15] S. K. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 1997.

[16] J. R. Quinlan. Learning with Continuous Classes. In *5th Australian Joint Conference on Artificial Intelligence*, pages 343–348, 1992.

[17] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufman, 1993.

[18] L. Torgo. Functional models for regression tree leaves. In *Proc. 14th International Conference on Machine Learning*, pages 385–393. Morgan Kaufmann, 1997.

[19] L. Torgo. Kernel regression trees. In *European Conference on Machine Learning*, 1997. Poster paper.

[20] L. Torgo. A comparative study of reliable error estimators for pruning regression trees. In H. Coelho, editor, *Iberoamerican Conference on Artificial Intelligence.* Springer-Verlag, 1998.

APPENDIX

A. PROOF OF PROPOSITION 1

For a Gaussian distribution with mean μ and covariance matrix $\Sigma = GG^T$, positive definite, and density $p_{\mu,\Sigma}(\mathbf{x})$ and a hyperplane with normal $\mathbf{n}$ that contains the point $\mathbf{x}_c$, the p-value with respect to the hyperplane is:

$$\begin{aligned} P[\mathbf{n}^T(\mathbf{x}-\mathbf{x}_c) \geq 0|\mu,\Sigma] &= \int_{\mathbf{n}^T(\mathbf{x}-\mathbf{x}_c)\geq 0} p_{\mu,\Sigma}(\mathbf{x})\mathrm{d}\mathbf{x} \\ &= \frac{\sigma}{2\sqrt{|\Sigma||S|}}\left(1+\mathrm{Erf}\left(\frac{\mu'_1}{\sigma\sqrt{2}}\right)\right) \end{aligned}$$

where

$$\Sigma'^{-1} = \begin{pmatrix} s & \mathbf{w}^T \\ \mathbf{w} & S \end{pmatrix} = M^T\Sigma^{-1}M$$

with M orthogonal such that $M^T\mathbf{n} = \mathbf{e}_1$, $\sigma = 1/\sqrt{s-\mathbf{w}^TS^{-1}\mathbf{w}}$ and $\mu' = M^T(\mu - \mathbf{x}_c)$.

PROOF. Since $M^T\mathbf{n} = \mathbf{e}_1$ the first column in M has to be $\mathbf{n}$ (which is supposed to have unitary norm) and the rest of columns are vectors orthogonal on $\mathbf{n}$. Such an orthogonal matrix can be found using Gram-Schmidth orthogonalization starting with $\mathbf{n}$ and the $d-1$ least paralel with $\mathbf{n}$ versor vectors. Doing the transformation $\mathbf{x}' = M^T(\mathbf{x}-\mathbf{x}_c)$ that transforms the hyperplane $\mathbf{n},\mathbf{x}_c$ into $\mathbf{e}_1, 0$ we get $\mathbf{x}-\mu = M(\mathbf{x}'-\mu')$. Using the notation Φ for $P[\mathbf{n}^T(\mathbf{x}-\mathbf{x}_c) \geq 0|\mu,\Sigma]$ and substituting in the definition we get:

$$\begin{aligned} \Phi &= \int_{x'_1\geq 0}\int_{x'_2}\cdots\int_{x'_d} p(\mathbf{x}')\mathrm{d}\mathbf{x}' \\ &= \int_{x'_1\geq 0}\int_{x'_2}\cdots\int_{x'_d} \frac{1}{(2\pi)^{d/2}\sqrt{|\Sigma|}} e^{-\frac{1}{2}[(\mathbf{x}'-\mu')^T\Sigma'^{-1}(\mathbf{x}'-\mu')]}\mathrm{d}\mathbf{x}' \end{aligned}$$

With the notation $\mathbf{y} = \mathbf{x}' - \mu'$ and L for the set of indeces $2\ldots d$, the exponent in the above integral can be rewritten like:

$$\begin{aligned} \mathbf{y}^T\Sigma'^{-1}\mathbf{y} &= sy_1^2 + 2y_1\mathbf{y}_L^Tw + \mathbf{y}_L^TS\mathbf{y}_L \\ &= sy_1^2 - y_1^2w^TS^{-1}w \\ &\quad + (\mathbf{y}_L + y_1S^{-1}w)^TS(\mathbf{y}_L + y_1S^{-1}w) \end{aligned}$$

With this we get:

$$\begin{aligned} \Phi_L(x'_1) &= \int_{\mathbf{x}'_L} \exp\left(-\frac{1}{2}[(\mathbf{x}'-\mu')^T\Sigma'^{-1}(\mathbf{x}'-\mu')]\right)\mathrm{d}\mathbf{x}'_L \\ &= \int_{\mathbf{y}_L} \exp\left(-\frac{1}{2}[sy_1^2 - y_1^2w^TS^{-1}w \right. \\ &\qquad \left. +(\mathbf{y}_L + y_1S^{-1}w)^TS(\mathbf{y}_L + y_1S^{-1}w)]\right)\mathrm{d}\mathbf{y}_L \\ &= \frac{(2\pi)^{\frac{d-1}{2}}}{\sqrt{|S|}} \exp\left(-\frac{1}{2}(x'_1-\mu'_1)^2(s-w^TS^{-1}w)\right) \end{aligned}$$

and substituting back in A we have:

$$\begin{aligned} \Phi &= \int_{x'_1\geq 0} \frac{1}{(2\pi)^{d/2}\sqrt{|\Sigma|}}\Phi_L(x'_1) \\ &= \frac{\sigma}{\sqrt{|\Sigma||S|}}\int_{x'_1\geq 0}\frac{1}{\sqrt{2\pi}\sigma}e^{-(x'_1-\mu'_1)^2/2\sigma^2} \\ &= \frac{\sigma}{\sqrt{|\Sigma||S|}}\int_{t\geq -\mu'_1/\sigma\sqrt{2}}\frac{1}{\sqrt{\pi}}e^{-t^2}\mathrm{d}t \\ &= \frac{\sigma}{\sqrt{|\Sigma||S|}}\left(\frac{1}{2}\frac{2}{\sqrt{\pi}}\int_{-\frac{\mu_1}{\sigma\sqrt{2}}}^{0} e^{-t^2}\mathrm{d}t + \frac{1}{\sqrt{\pi}}\int_0^\infty e^{-t^2}\mathrm{d}t\right) \\ &= \frac{\sigma}{2\sqrt{|\Sigma||S|}}\left(1+\mathrm{Erf}\left(\frac{\mu'_1}{\sigma\sqrt{2}}\right)\right) \end{aligned}$$

We show now that $s - \mathbf{w}^TS^{-1}\mathbf{w} > 0$ thus the above computations are sound. Since Σ is positive definite by supposition, $\Sigma'^{-1} = M^T\Sigma^{-1}M$ is positive definite. This means that $\mathbf{v}^T\Sigma'^{-1}\mathbf{v} > 0$ for any nonzero $\mathbf{v}$. Taking $\mathbf{v} = [1\ S^{-1}\mathbf{w}]^T$ we get the required inequality. $\square$