Classification and Regression Tree Construction

Thesis Proposal

Alin Dobra
Department of Computer Science
Cornell University, Ithaca NY
dobra@cs.cornell.edu
November 25, 2002

1 Introduction

Decision trees, either classification or regression trees, are especially attractive type of models for three main reasons. First, they have an intuitive representation, the resulting model is easy to understand and assimilate by humans [BFOS84]. Second, the decision trees are nonparametric models, no intervention being required from the user, and thus they are very suited for exploratory knowledge discovery. Third, scalable algorithms, in the sense that the performance degrades gracefully with the increase of the size of training data, exist for decision tree construction models [GRG98]. Last, accuracy of decision trees is comparable or superior to other models [Mur95, LLS97].

Three subtopics of decision tree construction received our attention:

- **Bias and Bias Correction in Split Selection Methods:** Preferences towards split variables with particular characteristics (i.e. bigger domains) can harm the interpretability and accuracy of decision trees. Methods to identify and correct such preferences are of significant interest.
- **Scalable Linear Regression Trees:** Linear regression trees are substantially more accurate than regression trees with constant regressors in the leaves but no scalable construction algorithms for such regressors have been proposed. There is great practical interest in a scalable algorithm to build such regression trees.
- **Probabilistic Decision Trees:** Decision trees are essentialy deterministic functions from the domain of predictor attributes to the domain of predicted attribute (class labels or real values). Some applications require probabilistic models or require the prediction of the model to be continous and differentiable. Probabilstic decision trees generalize decision trees and have these properties. Due to the decision tree heritage, they can be constructed efficiently.

In Section 2 we give some background information on classification and regression trees and the EM algorithm for Gaussian mixtures. In Section 3 we present our ongoing and future work on classification and regression tree construction. Section 3.1 contains details on our work on the bias correction in decision tree construction, followed by scalable linear regression tree construction in Section 3.2 and probabilistic classification and regression trees in Section 3.3. Section **??** contains some details on other ongoing work we are conducting on query processing over streaming data.

We conclude in Section 4 with a summary of past and proposed future work that constitutes our thesis proposal.

2 Preliminaries

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

2.1 Classification Trees

Let $X_1, \ldots, X_m, C$ be random variables where X_i has domain $\texttt{Dom}(X_i)$. The random variable C has domain $\texttt{Dom}(C) = \{1, \ldots, J\}$. We call $X_1 \ldots X_m$ *predictor attributes* (m is the number of predictor attributes) and C the *class label*.

A *classifier* $\mathcal{C}$ is a function $\mathcal{C} : \texttt{Dom}(X_1) \times \cdots \times \texttt{Dom}(X_m) \mapsto \texttt{Dom}(C)$. Let $\Omega = \texttt{Dom}(X_1) \times \cdots \times \texttt{Dom}(X_m) \times \texttt{Dom}(C)$ (the set of events).

For a given classifier $\mathcal{C}$ and a given probability measure P over Ω we can introduce a functional $R_P(\mathcal{C}) = P[\mathcal{C}(X_1, \ldots, X_n) \neq C]$ called the *misclassification rate* of the classifier $\mathcal{C}$.

Classifier Construction Problem: Given a training dataset D of N independent identically distributed samples from Ω, sampled according to probability distribution P, find a function $\mathcal{C}$ that minimizes the functional $R_P(\mathcal{C})$, where P is the probability distribution used to generate D.

A special type of classifier is a *decision tree*. A decision tree is a directed, acyclic graph $\mathcal{T}$ in the form of a tree. 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_Y on the outgoing edges of an internal node T must be *non-overlapping* and *exhaustive*. A set of predicates Q is *non-overlapping* if the conjunction of any two predicates in Q evaluates to `false`. A set of predicates Q is *exhaustive* if the disjunction of all predicates in Q evaluates to `true`. We will call the set of predicates Q_T on the outgoing edges of an internal node T the *splitting predicates of* T; the combined information of splitting attribute and splitting predicates is called the *splitting criteria* of T and is denoted by $\texttt{crit}(T)$.

Given a decision tree $\mathcal{T}$, we can define the associated classifier in the following recursive manner:

$$\mathcal{C}(x_1, \ldots, x_m, T) = \begin{cases} \text{Label}(T) & \text{if } T \text{ is a leaf node} \\ \mathcal{C}(x_1, \ldots, x_m, T_j) & \text{if } T \text{ is an internal node, } X_i \text{ is label of } T, \text{ and} \\ & q_{(T,T_j)}(x_i) = \texttt{true} \end{cases} \quad (1)$$

$$\mathcal{D}_{\mathcal{T}}(x_1, \ldots, x_m) = \mathcal{C}(x_1, \ldots, x_m, \text{Root}(\mathcal{T})) \quad (2)$$

If the tree $\mathcal{T}$ is a well-formed decision tree (as defined above), then the function $\mathcal{D}_{\mathcal{T}}()$ is also well defined and by our definition a classifier which we call a *decision tree classifier*, or short a *classification tree*. We can now formally state the problem of classification tree construction:

Classification Tree Construction Problem: Given a dataset $D = \{\omega_1, \ldots, \omega_N\}$ where the ω_i are independent identically distributed random samples from a probability distribution P over Ω, find a classification tree $\mathcal{T}$ such that the misclassification rate functional $R_P(\mathcal{D}_{\mathcal{T}})$ is minimized.

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 note 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

Figure 2 is an example decision tree constructed from the training database shown in Figure 3.

A classification tree is usually constructed in two phases [BFOS84]. 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 an approximation of $R_P(\mathcal{D}_\mathcal{T})$. A survey of the pruning methods can be found in [Mur97]. They are not very relevant for the work we are presenting so we are omitting the details. Most classification tree construction algorithms grow the tree top-down in the greedy way [BFOS84] shown in Figure 1. Note that the algorithm shown in Figure 1 takes a split selection method as argument.

We mention here the most popular split selection methods. Impurity based split selection methods form a distinct class. A discussion of the properties this split selection methods have can be found in Breiman et al.([BFOS84]).

Gini Index. This popular split selection method has been used widely in previous work on classification tree construction in the database literature such as SLIQ [MAR96], SPRINT [SAM96], and PUBLIC [RS98]. The value of the gini index is defined as follows:

$$gini(T) \stackrel{\text{def}}{=} 1 - \sum_{i=1}^{J} P[i|T], \quad GG(T, X, Q) \stackrel{\text{def}}{=} gini(T) - \sum_{j=1}^{n} P[q_j(X)|T] \cdot gini(T_j) \tag{3}$$

Information Gain. This popular split selection method is widely used in the machine learning literature [Mit97].

$$entropy(T) \stackrel{\text{def}}{=} -\sum_{i=1}^{J} \log P[i|T], IG(T, X, Q) \stackrel{\text{def}}{=} entropy(T) - \sum_{j=1}^{n} P[q_j(X)|T] \cdot entropy(T_j) \tag{4}$$

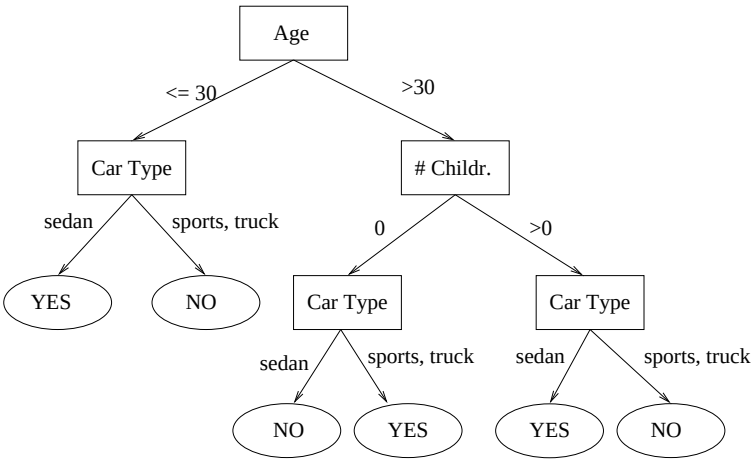

Figure 2: Example of decision tree

Car type	Age	Chd	Sub
sedan	23	0	yes
sports	31	1	no
sedan	36	1	no
truck	25	2	no
sports	30	0	no
sedan	36	0	no
sedan	25	0	yes
truck	36	1	no
sedan	30	2	yes
sedan	31	1	yes
sports	25	0	no
sedan	45	1	yes
sports	23	2	no
truck	45	0	yes

Figure 3: Example Training Database

Gain Ratio. Quinlan introduced this adjusted version of the information gain to counteract the bias introduced by categorical attributes with large domains [Qui86].

$$GR(T,X,Q) \stackrel{\text{def}}{=} \frac{IG(T,X,Q)}{-\sum_{j=1}^{|\mathtt{Dom}(X)|} \log P[X=x_j|T]} \tag{5}$$

There are two other popular split selection methods for n-ary splits for categorical attributes that are not based on an impurity function.

The χ^2 Statistic.

$$\chi^2(T) \stackrel{\text{def}}{=} \sum_{i=1}^{J} \sum_{j=1}^{|\mathtt{Dom}(X)|} \frac{(P[X=x_i|T]\cdot P[C=j|T] - P[X=x_i, C=j|T])^2}{P[X=x_i|T]\cdot P[C=j|T]}. \tag{6}$$

The G Statistic.

$$G(T) \stackrel{\text{def}}{=} 2\cdot N\cdot IG(T)\log_e 2, \tag{7}$$

where N is the number of records at node T.

2.2 Regression Trees

We start with the formal definition of the regression problem and we present regression trees, a particular type of regressor.

We have the random variables $X_1,\ldots,X_m$ as in the previous section to which we add the random variable Y with real line as the domain that we call the *predicted attribute* or output.

A *regressor* $\mathcal{R}$ is a function $\mathcal{R} : \mathtt{Dom}(X_1)\times\cdots\times\mathtt{Dom}(X_m)\mapsto\mathtt{Dom}(Y)$. Now if we let $Omega = \mathtt{Dom}(X_1)\times\cdots\times\mathtt{Dom}(X_m)\times\mathtt{Dom}(Y)$ we can define probability measures P over Ω. Using such a

probability measure and some loss function $\mathcal{L}$ (i.e. square loss function $\mathcal{L}(a,x) = \|a-x\|^2$) we can define the regressor error as $R_P(\mathcal{R}) = E_P[\mathcal{L}(Y, \mathcal{R}(X_1, \ldots, X_m)]$ where E_P is the expectation with respect to probability measure P.

Regressor Construction Problem: Given a training dataset D of N iid samples from Ω sampled according to probability distribution P, find a function $\mathcal{R}$ that minimizes the functional $R_P(\mathcal{R})$.

Regression Trees, the particular type of regressors we are interested in, are the natural generalization of decision trees for regression (continuous valued prediction) problems. Instead of associating a class label to every node, a real value or a functional dependency of some of the inputs is used.

Regression trees were introduced by Breiman et al.(CART) [BFOS84]. Regression trees in CART have a constant numerical value in the leaves and use the variance as a measure of impurity. Thus the split selection measure is:

$$\mathrm{Err}(T) \stackrel{\mathrm{def}}{=} \frac{1}{N_T}\sum_{i=1}^{N_T}(y_i - \overline{y_i})^2, \quad \Delta\mathrm{Err}(T) = \mathrm{Err}(T) - \sum_{j=1}^{n} P[q_j(X)|T] \cdot \mathrm{Err}(T_j) \tag{8}$$

The reason for using variance as the impurity measure is justified by the fact that the best constant predictor in a node is the average of the value of the predicted variable on the test examples that correspond to the node; the variance is thus the mean square error of the average used as a predictor.

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

Usually the top-down induction schema algorithm like the one in Figure 1 is used to build regeression tress. Pruning is used to improve the accuracy on unseen examples like in the classification tree case. Pruning methods for classification trees can be adapted for regression trees [Tor98].

2.3 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. An iterative algorithm to find parameters of such Gaussian distributions was introduced by Wolfe [Wol70]) and was was latter generalized by Dempster et al. [DR77]. Our introduction follows mostly the excellent tutorial [Bil] where complete proofs 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:

$$p(\mathbf{x}, \Theta) = \sum_{i=1}^{M} \alpha_i p_i(\mathbf{x}|\mu_i, \Sigma_i) \tag{9}$$

$$p_i(\mathbf{x}|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu_i)^T\Sigma_i^{-1}(\mathbf{x}-\mu_i)} \tag{10}$$

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 with the normalization property $\sum_{i=1}^{M} \alpha_i = 1$. 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):

$$h_{ji} = \frac{\alpha_i p_i(\mathbf{x_j}|\mu_i, \Sigma_i)}{\sum_{k=1}^{M} \alpha_k p_k(\mathbf{x_j}|\mu_k, \Sigma_k)} \tag{11}$$

Maximization (M step):

$$\alpha_i = \frac{1}{N} \sum_{j=1}^{N} h_{ij} \tag{12}$$

$$\mu_i = \frac{\sum_{j=1}^{N} h_{ij} \mathbf{x}_j}{\sum_{j=1}^{N} h_{ij}} \tag{13}$$

$$\Sigma_i = \frac{\sum_{j=1}^{N} h_{ij} (\mathbf{x}_j - \mu_i)(\mathbf{x}_j - \mu_i)^T}{\sum_{j=1}^{N} h_{ij}} \tag{14}$$

It follows immediately from the above equations that Σ_i is a positive definite matrix if vectors $\mathbf{x}_j$ are linear independent.

3 Classification and Regression Tree Construction

As mentioned in the introductory part most of our work adresses issues in the decision and regression tree construction. Even though this space has been thoroughly explored for the last two decades, it still provides interesting research oportunities.

3.1 Bias Correction in Decision Tree Construction

Split variable selection is one of the main components of classification tree construction. The quality of the split selection criterion has a major impact on the quality (generalization, interpretability and accuracy) of the resulting tree. Many popular split criteria suffer from bias towards predictor variables with large domains [WL94, Kon95].

Consider two predictor variables X_1 and X_2 whose association with the class label is equally strong (or weak). Intuitively, a split selection criterion is unbiased if on a random instance the criterion chooses both X_1 and X_2 with probability $1/2$ as split variables. Unfortunately, this is usually not the case.

There are two previous meanings associated with the notion of *bias* in decision tree construction: an anomaly observed by [Qui86], and the difference in distribution of the split criteria applied to different predictor variables [WL94]. The first is very specific and doesn't extent to the intuitive definition given above, the second one is too strict in the sense that it requires equality of distributions, which is hard to check.

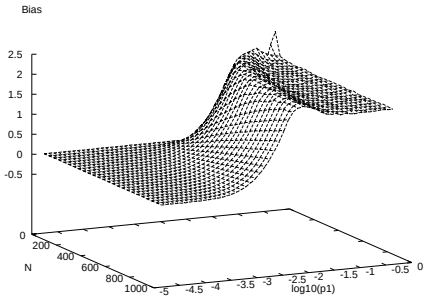

Figure 4: The bias of the information gain.

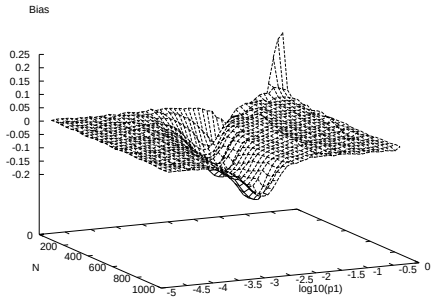

Figure 5: The bias of the p-value of the χ^2-test (using a χ^2-distribution).

In our work we defined bias of a split criterion s as the logarithmic odds of choosing X_1, a split attribute, over X_2, some other split attribute, when neither X_1 nor X_2 is correlated with the class label [DG01]. Formally:

$$\text{Bias}(X_1, X_2) = \log_{10}\left(\frac{P_s(X_1, X_2)}{1 - P_s(X_1, X_2)}\right) \tag{15}$$

where

$$P_s(X_1, X_2) \stackrel{\text{def}}{=} P[s(\mathcal{D}, X_1) > s(\mathcal{D}, X_2)] \tag{16}$$

is the probability that split criterion s chooses X_1 over X_2. $\mathcal{D}$ is a dataset distributed according to the hypothesis that the input attributes are not corelated with the class label (Null Hypothesis).

When the split criterion is unbiased $Bias(X_1, X_2) = \log_{10}(0.5/(1-0.5)) = 0$. The bias is positive if s prefers X_1 over X_2 and negative, otherwise. Instead of specifying when a split criterion is biased or not, this definition gives a measure of the bias. This is important since in practice we are interested in almost unbiased split criteria, not necessarily strictly unbiased.

Using this definition of the bias we showed that a number of popular split criteria have a pronounced bias towards split attributes with big domains [DG01]. Figure 4 depicts the bias of the information gain split criterion as a function of the number of training examples N and probability to see the first class label p_1. As can be easily noticed, for values of p_1 between 10^{-2} and $1/2$ there is a very strong bias towards predictor attibute X_2 that has 10 possible values (predictor attribute X_2 has 2 possible values). Similar results were observed for gini gain and gain ratio (but with smaller bias). Figure 5 is the result of the same experiment for the χ^2 test. For this criterion the bias is almost nonexistent. The same qualitative result can be observed for G^2 statistic but with more noticeable bias.

These experimental results suggest that there is something inherent in the nature of the statistical split criteria that makes them have a much smaller biase than the non-statistical criteria. We were able to prove that the p-value of any split selection criterion (the probability, under the Null Hypothesis, that a value of the criterion as good or better than the one observed is obtained) is virtually unbiased [DG01].

Thus, to get an unbiased criterion, the p-value of any split criterion can be used. The only difficulty in this method is the computation of the p-value under the Null Hypothesis of a given criterion. We can distinguish four ways in which this can be accomplished:

- Exact computation. Use the exact distribution of the split criterion. The main drawback is that this is almost always very expensive; it is reasonably efficient only for $n = 2$ and $k = 2$ [Mar97].
- Bootstrapping. Use Monte Carlo simulations with random instances generated according to the Null Hypothesis. This method was used in by [FW98]; its main drawback is the high cost of the the Monte Carlo simulations.
- Asymptotic approximations. Use an asymptotic approximation of the distribution of the split criterion (e.g., use the χ^2-distribution to approximate the χ^2-test [Kas80] and the G^2-statistic [Min87]). Approximations often work well in practice, but they can be inaccurate for border conditions (e.g., small entries in the contingency table).
- Tight approximations. Use a tight approximation of the distribution of the criterion with a *nice* distribution. While conceptually superior to the previous three methods, such tight approximations might be hard to find.

Out of the four possible ways to compute the p-value of a split criterion the last one, tight approximation, is the most appealing since is both effective and computationally efficient. We were able to deduce such a tight approximation for the gini index for decision trees that have an outgoing branch for intermediate nodes for every possible value of the split attribute. The key ideas of the approximation are: (1) the experimental observation that the distribution of the gini index under the Null Hypothesis is essentially a gamma distribution and (2) the expected value and variance of this split criterion can be computed exactly. Thus, the distribution of the gini index is approximated by a gamma distribution with the same expected value and variance (two parameters completely determine a gamma distribution). Details and actual formulas can be found in [DG01]. Experiments with this criterion revealed a behavior similar with the one in Figure 5.

Summary: Our prior work on the problem of bias in split selection methods has the following three contributions:

1. We defined *Bias* is split selection as the log odds of choosing a split variable over another. This defines a useful measure that captures the intuitive notion of bias. A mathematical definition of the bias is crucial for any theoretical developments.
2. We proved that the p-value of *any* split selection criterion is essentially unbiased, thus giving a very general method to obtain unbiased criteria. This result also explains why the statistical criteria previously used are unbiased.
3. We constructed a *tight approximation* of the distribution of gini gain split criterion. We used this distribution to build an unbiased criterion following our general prescription. The distribution of gini gain is interesting in itself since it can be used for pre or post pruning much in the way the χ^2 test is used [Fra00].

Future Work and Research Directions: The work we just presented on the bias in split selection is only an initial step. A thorough theoretical and experimental study of the proposed correction of gini index is in order. We have done some progress in the direction of estimating the distribution of the gini index for general situations, not only under Null Hypothesis. Determining the distribution is crucial for a theoretical characterization of the correction.

3.2 Scalable Linear Regression Trees

We introduced the regression trees in Section 2.2. We mention here only some further developments that are related to our work.

Even though they were introduced early in the development of decision trees (CART, Breiman et al. [BFOS84]), they received far less attention from the research community. Quinlan generalized the regression trees in CART by using a linear model in the leaves to improve the accuracy of the prediction [Qui92]. The impurity measure used to choose the split variable and the split point is the standard deviation of the predictor of the training examples at the node. Karalilc argues that the mean square error of the linear model in a node is a more appropriate impurity measure for the linear regression trees since it is possible to have data with big variance but well predicted by a linear model [Kar92]. This is a crucial observation since evaluating the variance is much easier than the error of a linear model (that requires solving a linear system). Even more, if discrete attributes are present among the predictor attributes and CART type of trees are built (at most two children for each node), 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 [BFOS84]) doesn't seem to apply.

Torgo proposed the usage of even more sophisticated functional models in the leaves (i.e. kernel regressors) [Tor97]. For such regression trees both construction and usage of the model is expensive but the accuracy is superior to the linear regression trees.

There are a number of contributions coming from the *Statistics* community. Chaudhuri et. al proposed the use of statistical tests for split variable selection instead of error of fit methods [CHLY94]. The main idea is to fit a model (constant, linear or higher order polynomial) in every node in the tree and to partition the data at this node into two classes: datapoints with positive residuals[1] 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 decision tree construction can be used from this point on. In this work Student's t-test was used. Unfortunately, it is not very clear why differences in the distributions of the signs of the residuals are good criteria on which decision about attributes to split on and split points are made. A further enhancement was proposed recently by Loh [Loh02]. It consists mostly in the use of 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.

Motivation: We are interested mostly in designing a scalable regression tree algorithm with linear regressors in the leaves. The choice of linear regression trees as model for scalable regression problem is motivated by the following properties of linear regression trees:

- Due to the hierarchical structure, regression trees are easy to understand and interpret.
- *Divide-et-impera*-like methods can be used to build regression trees, thus the computational effort can be made linear in the size of the training data and logarithmic in the size of the model. This is the key for scalable regression methods.
- Regression trees, especially with linear models in the leaves, are capable of achieving great

[1] Residuals are the difference between the true value and the value predicted by regression model.

accuracy and the models are non-parametric (there is no apriori limit on the number of parameters).

- Predictions using a regression tree model can be made very efficiently (proportional to the logarithm of the size of the model).

For constant regression trees, algorithms for scalable classification trees can be straightforwardly adapted [GRG98]. 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 $A^T A$ that defines the linear system). This can be a problem also.

We make the distinction in [Loh02] between predictor attributes:

- *Discrete attributes* used only in the split predicates in intermediate nodes in the regression tree.
- *Split continuous attributes*: continuous attributes used only for splitting.
- *Regression attributes*: continuous attributes used in the linear combination that specifies the linear regressors in the leaves as well as for specifying split predicates.

Proposed Solution: The main idea is to locally transform the regression problem into a classification problem. This can be done by first identifying two general Gaussian distributions in the regressor attributes – output space using EM algorithm for Gaussian mixtures and then classifying the datapoints based on the probability to belong to these two distributions. Classification tree techniques are then used to select the split attribute and the split point.

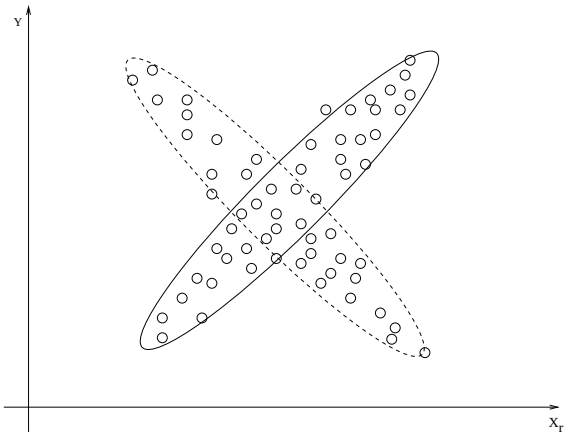

Figure 6: Projection on X_r, Y space of traning data.

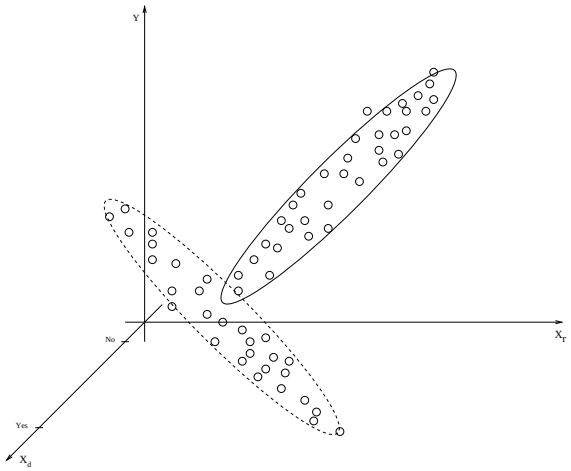

Figure 7: Projection on X_d, X_r, Y space of same training data as in Figure 6

The role of EM is to identity two natural classes in the data that have approximatively a linear organization. The role of classification step is to identify predictor attributes that can make the

difference between these two classes in the input space. Too see this more clearly suppose we are in the process of building a linear regression tree. Suppose that we have a hypothetical set of training examples 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 6. The data is 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. The information in the discrete attribute X_d can make this distinction, as can be observed from Figure 7 where the projection is made on the space X_d, X_r, Y. If more split attributes would have been present, a split on X_d would have been preferred since the classes after split are pure.

Observe that use of EM algorithm for Gaussian mixture 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 EM algorithm with clusters initialized randomly works well.

Benefits of Proposed Solution:

- The regression algorithm is very intuitive and simple since it consists mainly of a straightforward combination of known and well studied techniques, namely: split selection in classification trees and EM algorithm for Gaussian mixtures.
- By employing scalable versions of variable selection in classification tree construction algorithms [GRG98] and EM algorithm for Gaussian mixtures [BFR98] a scalable linear regression tree construction algorithms can be built.
- Good oblique splits[2] can be found for regression trees build with our algorithm. The projections of the datapoints corresponding to the two classes, induced by the two Gaussian clusters, on the continuous predictor attributes are approximated with two Gaussian distributions (one for each class). The hyperplane that best separates this distributions determines the *optimal* oblique split. The construction of this oblique splits incurs minimum computation overhead and can substantially improve the interpretability and accuracy of the regression trees.

Efficient Implementation of EM Algorithm The following two ideas are used to implement efficiently the EM algorithm:

- Steps E and M are performed simultaneously. This means that quantities h_{ij} do not have to be stored explicitly.
- All the operations are expresed 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 diagonal, so solving linear systems takes quadratic effort in the number of dimensions and computing the determinant is linear in the number of dimensions.

[2]Oblique splits are inequalities involving linear combinations of split and regression continuous attributes.

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 10 we get:

$$p_i(\mathbf{x}|\mu_i, G_i) = \frac{1}{(2\pi)^{d/2}|G_i|} e^{-\frac{1}{2}\|G_i^{-1}(\mathbf{x}-\mu_i)\|^2} \tag{17}$$

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:

$$s_i = \sum_{j=1}^{N} h_{ij} \tag{18}$$

$$s_{\mathbf{x},i} = \sum_{j=1}^{N} h_{ij}\mathbf{x}_j \tag{19}$$

$$S_i = \sum_{j=1}^{N} h_{ij}\mathbf{x}_j^T\mathbf{x}_j \tag{20}$$

and for every training example $\mathbf{x}_j$ quantities h_{ij} are computed with the formula in Equation 11. 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 M distributions are computed with the formulas:

$$\alpha_i = \frac{s_i}{N} \tag{21}$$

$$\mu_i = \frac{s_{\mathbf{x},i}}{s_i} \tag{22}$$

$$\Sigma_i = \frac{S_i}{s_i} - \mu_{\mathbf{i}}{}^T\mu_{\mathbf{i}} \tag{23}$$

$$G_i = \mathrm{Chol}(\Sigma_i) \tag{24}$$

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.

The Gaussian mixture problem is solved in the regressor–output space and for each mixture we have to find the least square estimate (LSE) of the output as a function of regressors. It can be shown that the LSE regressor is:

$$y = \mathbf{c}_i(\mathbf{x} - \mu_{I,i}) + \mu_{O,i} \tag{25}$$

where $\mathbf{c}_i$ is the solution of linear equation $\mathbf{c}_i^T G_{II,i} = G_{OI,i}$. Subscript I was used to denote the first $d-1$ components (the ones referring to regressors) and O to refer to the last component (corresponds to output). Thus, for example $G_{II,i}$ refers to the $(d-1)\times(d-1)$ upper part of G_i.

Prior Work: We have fully implemented the algorithm described above for construction of linear regression trees. The only pruning method implemented is Quinlan's resubstitution error pruning.

We conducted some preliminary experiments in which we learned smooth nonlinear three dimensional functions with and without normal noise added. On these experiments we observed improvements of up to 4 in accuracy on resulted trees with up to 20% less nodes when compared with GUIDE regression tree learner [Loh02].

Future Work:

- Implement more pruning methods, for example Complexity Pruning.
- Conduct extensive experiments on synthetic and real life datasets of the proposed regression method, comparing it with other well established regression methods in terms of accuracy, size of the resulting model and computational effort.

3.3 Probabilistic Decision and Regression Trees

Decision and regression trees prove to be an excellent class of models for practical applications. Nevertheless they have a number of shortcomings for some applications:

- Only one path in the tree is used to make predictions. This is the case since any path in the tree defines a partition of the input space and a particular value of the input can fall only in one such partition. This has as consequence a lack of precision of the prediction when the input is close to the border of the partition. For regression trees it also means that the model seen as a function has discontinuities so it does not have a derivative everywhere.
- In the learning process the training examples influence the construction of subparts of the tree. After a split strategy is chosen, only training examples that satisfy the split predicate are used to construct a child node, thus information contained in a datapoint is used only in one child. This leads to a severe reduction in the number of datapoints on which decisions are made as the tree construction progresses and has as consequence a reduction in the precisions of the estimates.
- Mistakes in the choice of variables to split on in an intermediate node result not only in increasing the size of the model, but have as consequence a decrease of precision of decisions due to data fragmentation.
- For some applications it is desirable for decision trees to give a probabilistic answer instead of the deterministic answer. Estimating such probabilities only from the information at the leaves incurs big errors due to data fragmentation.

In order to address these fallacies we are proposing a generalization of the decision and regression trees that makes extensive use of probability distributions. To simplify the exposition, we first present probabilistic decision trees (PDT) and just indicate afterwards the way probabilistic regression trees differ.

First we introduce some notation. Let k be the number of class labels (size of the domain of the predicted variable). Let x be the current input (vector with values for all the input variables).

Like decision trees, PDTs consist of a set of nodes that are linked in a tree structure. We have two types of nodes:

Intermediate Nodes: are nodes with exactly k children. For each child a function $f_{T,i}$ defined on the space of all the inputs is specified. This function is interpreted as the density of a probability distribution and its application to input x gives the probability that x influences branch i of the subtree starting at the current node. Functions $f_{T,i}$ satisfy the normalization condition $\sum_{i=1}^{k} f_{T,i}(x) = 1$ for any input x. The vertex between nodes T and T_i is labeled by the function $f_{T,i}$.

Leaf Nodes: are nodes without children. The only information attached to them is the predicted class label.

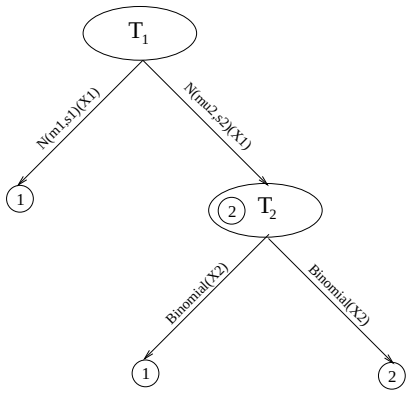

Figure 8: Example of probabilistic decision tree.

An example of PDT is depicted in Figure 8. There are two class labels 1 and 2 so each of the intermediate nodes T_1 and T_2 has two children. For the leaves we represented only the class label. For node T_1 the two functions that induce the probability to get to each of the children are densities of normal distributions on predictor variable X_1 (which is supposed continuous). For T_2 the two functions are densities of binomial distributions on the predictor variable X_2 supposed to be discrete.

Since function $f_{T,i}(x)$ is interpreted as the probability that input x goes on branch i at node T, the product of the applications to input x of the functions that label vertices on a path from the root node T_R to a leaf gives the probability to reach this particular leaf. By summing up the probabilities to reach labels with class label c we obtain the probability that the output has class label c. This can be done for all the class labels and thus a probability distribution of the prediction is obtained. If a deterministic prediction is required, the class label with the highest probability can be returned.

If functions $f_{T,i}$ for any node T are chosen so that they have value 1 if x is in the partition P_i and 0 otherwise and partitions P_i are simple in the sense that they refer to only one component of x, the PDTs degenerate into normal decision trees.

Learning PDTs: The main idea in constructing the functions $f_{T,i}$ for a node T during the learning of PDTs is to use all the training examples available but to weight them by the product of the applications of the functions on the path from the root node to the current node to the input part

of these datapoints. Thus the already constructed part of the PDT defines a *magnifying glass* that changes the importance of each of the training examples in the construction of the current node.

Since training examples are labeled for each child i a distribution that approximates well the distribution of the weighted datapoints with label i is determined. Only particular classes of parametric distributions are considered. The choice of the parameters of a distribution when the class is known is equivalent to the split point selection problem in decision tree construction. The class choice is equivalent with the split attribute selection problem. Examples of classes of distributions are: multinomial distributions on discrete predictor attributes, normal distributions on continuous predictor attributes; one class for each predictor attribute.

Definition of impurity measures like gini index and information gain use prior and conditional probabilities only, thus they can also be used to select a class of distributions to use in a node of PDTs.

Probabilistic Regression Trees (PRT): They differ from PDTs only in the fact that regression models (i.e. constant, linear) label the leaves instead of class labels. The prediction of PRTs is the weighted sum of the predictions of regressors in the leaves, the weight of a leaf being the probability to reach the leaf for the particular input from the root node. We see no reason to have more than two children for a node in a PRTs. The technique described in Section 3.2 can be used to locally reduce the regression problem to a classification problem.

Future work: Some of the ideas mentioned above require further refinements. Also pruning of PDTs and PRTs has to be considered. Once the learning algorithms are perfected we plan to do an extensive empirical evaluation with synthetic and real life data sets.

Related work: In terms of arhitecture our proposal is closed to hierarchical mixture of experts [JJ93, WR94] and bayesian network classifiers [FG96]. One particular technique proposed to acheive continuity in the resulting decision tree models is smoothing by averaging [CHLY94, Bun93]. By fitting a naive bayesian classifier in the leaves of a decision tree, probabilistic predictions can be made [BFOS84]. This types of models are usually called class probability trees. Buntine combined smoothing by averaging with class probability trees [Bun93].

4 Summary of Thesis Proposal

We now summarize our present and future work on classification and regression tree construction:

- **Bias in Split Selection:** We defined *bias* in split selection as the log adds of choosing a split attribute over another. Using this definition we showed that popular split selection criteria have significant bias. We then proved that the *p-value* of any split selection criteria is essentially an unbiased split criteria and we used this result to correct the bias of gini index. To build such a correction we found a good approximation of the distribution under *Null Hypothesis* of gini index. We will continue this work with a theoretical and experimental study of the proposed correction on gini index.
- **Scalable Linear Regression Trees:** We proposed a novel method to learn regression trees with linear regressors in the leaves. The method consists in determining at the level of every node during tree construction, using EM algorithm for Gaussian mixtures, two Gaussian

clusters with general covariance matrices that fit well the training data. The clusters are then used to classify the training data in two classes based on closeness to this clusters. In this way a classification problem results for each node and classification tree construction techniques can be used to make good decisions. A scalable version of the proposed algorithm can be obtained by employing scalable EM and classification tree construction algorithms. We implemented the proposed method with resubstitution error pruning and obtained some good initial results. We plan to implement more pruning methods and to conduct a detailed experimental study.

- **Probabilistic Decision Trees:** We proposed a generalization of classification and regression trees that consists in using probability distributions instead of split predicates for intermediate nodes. Some of the benefits of such a model are: probabilistic predictions are made in a natural way, unnecessary fragmentation of the training data is avoided, resulted models have continuity properties, more insights can be gained from a probabilistic modes. We proposed inference and learning methods for such trees. We are planing to adapt decision tree pruning methods to these types of models and scalable versions of the construction algorithms.

References

[BFOS84] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, 1984.

[BFR98] P. S. Bradley, Usama M. Fayyad, and Cory Reina. Scaling clustering algorithms to large databases. In *Knowledge Discovery and Data Mining*, pages 9–15, 1998.

[Bil] Jeff Bilmes. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models.

[Bun93] W. Buntine. Learning classification trees. In D. J. Hand, editor, *Artificial Intelligence frontiers in statistics*, pages 182–201. Chapman & Hall,London, 1993.

[CHLY94] P. Chaudhuri, M.-C. Huang, W.-Y. Loh, and R. Yao. Piecewise-polynomial regression trees. *Statistica Sinica*, 4:143–167, 1994.

[DG01] Alin Dobra and Johannes Gehrke. Bias correction in classification tree construction. In *Proceedings of the Eighteen International Cornerence on Machine Learning*, pages 90–97. Morgan Kaufmann, 2001.

[DR77] 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.

[FG96] Nir Friedman and Moises Goldszmidt. Building classifiers using bayesian networks. In *AAAI/IAAI, Vol. 2*, pages 1277–1284, 1996.

[Fra00] Eibe Frank. *Pruning decision trees and lists*. PhD thesis, Department of Computer Science, University of Waikato, Hamilton, New Zealand, 2000.

[FW98] Eibe Frank and Ian H. Witten. Using a permutation test for attribute selection in decision trees. In *International Conference on Machine Learning*, 1998.

[GRG98] Johannes Gehrke, Raghu Ramakrishnan, and Venkatesh 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.

[JJ93] Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. Technical Report AIM-1440, 1993.

[Kar92] A. Karalic. Linear regression in regression tree leaves, 1992.

[Kas80] G. Kass. An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, (29):119–127, 1980.

[Kon95] I. Kononenko. On biases in estimating multivalued attributes, 1995.

[LLS97] Tjen-Sien Lim, Wei-Yin Loh, and Yu-Shan Shih. An empirical comparison of decision trees and other classification methods. Technical Report 979, Department of Statistics, University of Wisconsin, Madison, June 1997.

[Loh02] Wei-Yin Loh. Regression trees with unbiased variable selection and interaction detection. *Statistica Sinica*, 2002. to appear.

[MAR96] Manish Mehta, Rakesh Agrawal, and Jorma Rissanen. SLIQ: A fast scalable classifier for data mining. In *Proc. of the Fifth Int'l Conference on Extending Database Technology (EDBT)*, Avignon, France, March 1996.

[Mar97] J. Kent Martin. An exact probability metric for decision tree splitting. *Machine Learning*, 28(2,3):257–291, 1997.

[Min87] John Mingers. Expert systems – rule induction with statistical data. *J. Opl. Res. Soc.*, 38(1):39–47, 1987.

[Mit97] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[Mur95] Sreerama K. Murthy. *On growing better decision trees from data*. PhD thesis, Department of Computer Science, Johns Hopkins University, Baltimore, Maryland, 1995.

[Mur97] Sreerama K. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 1997.

[Qui86] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Qui92] J. R. Quinlan. Learning with Continuous Classes. In *5th Australian Joint Conference on Artificial Intelligence*, pages 343–348, 1992.

[RS98] Rajeev Rastogi and Kyuseok Shim. PUBLIC: A decision tree classifier that integrates building and pruning. In *Proceedings of the 24th International Conference on Very Large Databases*, pages 404–415, New York City, New York, August 24-27 1998.

[SAM96] John Shafer, Rakesh Agrawal, and Manish Mehta. SPRINT: A scalable parallel classifier for data mining. In *Proc. of the 22nd Int'l Conference on Very Large Databases*, Bombay, India, September 1996.

[Tor97] Luís Torgo. Functional models for regression tree leaves. In *Proc. 14th International Conference on Machine Learning*, pages 385–393. Morgan Kaufmann, 1997.

[Tor98] L. Torgo. A comparative study of reliable error estimators for pruning regression trees, 1998.

[WL94] Allan P. White and Whei Zong Liu. Bias in information-based measures in decision tree induction. *Machine Learning*, 15:321–329, 1994.

[Wol70] John H. Wolfe. Pattern clustering by multivariate mixture analysis. *multivariate behavioral research*, 5:329–350, July 1970.

[WR94] S. R. Waterhouse and A. J. Robinson. Classification using hierarchical mixtures of experts. In *Proceedings of the 1994 IEEE Workshop on Neural Networks for Signal Processing IV*, pages 177–186, Long Beach, CA, 1994. IEEE Press.