Machine Learning, 55, 219–250, 2004

Functional Trees

JOÃO GAMA jgama@liacc.up.pt
LIACC, FEP—University of Porto, Rua Campo Alegre 823, 4150 Porto, Portugal

Editor: Foster Provost

Abstract. In the context of classification problems, algorithms that generate multivariate trees are able to explore multiple representation languages by using decision tests based on a combination of attributes. In the regression setting, *model trees* algorithms explore multiple representation languages but using linear models at leaf nodes. In this work we study the effects of using combinations of attributes at decision nodes, leaf nodes, or both nodes and leaves in regression and classification tree learning. In order to study the use of functional nodes at different places and for different types of modeling, we introduce a simple unifying framework for multivariate tree learning. This framework combines a univariate decision tree with a linear function by means of constructive induction. Decision trees derived from the framework are able to use decision nodes with multivariate tests, and leaf nodes that make predictions using linear functions. Multivariate decision nodes are built when growing the tree, while functional leaves are built when pruning the tree. We experimentally evaluate a univariate tree, a multivariate tree using linear combinations at inner and leaf nodes, and two simplified versions restricting linear combinations to inner nodes and leaves. The experimental evaluation shows that all functional trees variants exhibit similar performance, with advantages in different datasets. In this study there is a marginal advantage of the full model. These results lead us to study the role of functional leaves and nodes. We use the bias-variance decomposition of the error, cluster analysis, and learning curves as tools for analysis. We observe that in the datasets under study and for classification and regression, the use of multivariate decision nodes has more impact in the bias component of the error, while the use of multivariate decision leaves has more impact in the variance component.

Keywords: multivariate decision trees, multiple models, supervised learning

1. Introduction

The generalization ability of a learning algorithm depends on the appropriateness of its representation language to express a generalization of the examples for the given task. Different learning algorithms employ different representations, search heuristics, evaluation functions, and search spaces. It is now commonly accepted that each algorithm has its own selective superiority (Brodley, 1995); each one is best for some but not all tasks. The design of algorithms that explore multiple representation languages and explore different search spaces has an intuitive appeal.

In the context of supervised learning problems it is useful to distinguish between classification problems and regression problems. In the former the target variable takes values in a finite and pre-defined set of un-ordered values, and the usual goal is to minimize a 0–1 loss function. In the latter the target variable is ordered and takes values in a subset of $\Re$. The usual goal is to minimize a squared error loss function. Mainly due to the differences in the type of the target variable successful techniques in one class of problems are not directly applicable to the other class of problems.

The supervised learning problem is to find an approximation to an unknown function given a set of labeled examples. To solve this problem, several methods have been presented in the literature. Two of the most representative methods are the *Generalized Linear Models* (Berthold & Hand, 1999) and *Decision trees* (Breiman et al., 1984; Quinlan, 1993a). These methods explore different hypothesis spaces and use different search strategies: they are appropriate for different types of problems (Perlich, Provost, & Simonoff, 2003). In the case of generalized linear models (GLM), the usual goal is to minimize the sum of squared deviations of the observed values for the dependent variable from those predicted by the model. It is based on the algebraic theory of invariants and has an analytical solution. The description language of the model takes the form of a polynomial that, in its simpler form, is a linear combination of the attributes: $w_0 + \sum w_i \times x_i$. This is the basic idea behind linear-regression and discriminant functions (McLachlan, 1992). In the case of decision trees a *divide-and-conquer* strategy is used. The goal is to decompose a complex problem into simpler problems and recursively to apply the same strategy to the sub-problems. Solutions of the sub-problems are combined in the form of a tree. Its hypothesis space is the set of all possible hyper-rectangular regions. The power of this approach comes from the ability to split the space of the attributes into subspaces, whereby each subspace is fitted with different functions. This is the basic idea behind well-known tree based algorithms like CART (Breiman et al., 1984) and C4.5 (Quinlan, 1993a).

In the case of classification problems, a class of algorithms that explore multiple representation languages are the so called *multivariate trees* (Breiman et al., 1984; Utgoff & Brodley, 1991; Murthy, Kasif, & Salzberg, 1994; Gama, 1997; Loh & Shih, 1997). In this sort of algorithm decision nodes can contain tests based on a combination of attributes. The language bias of univariate decision trees (axis-parallel splits) is relaxed allowing decision surfaces oblique with respect to the axes of the instance space. As in the case of classification problems, in regression problems some authors have studied the use of regression trees that explore multiple representation languages, here called *model trees* (Quinlan, 1992; Witten & Frank, 2000; Torgo, 2000). While in classification problems usually multivariate decisions appear in internal nodes, in regression problems multivariate decisions appear in leaf nodes. This observation is the main motivation for this work. Should we restrict combinations of attributes to decision nodes? Should we restrict combinations of attributes to leaf nodes? Would it be beneficial to use combinations of attributes both at decision nodes and leaf nodes?

We present a framework that allows us to analyze and discuss *where* should we use linear combinations of attributes. This framework is an extension of multivariate and model trees. It is applicable to regression and classification domains, allowing combinations of attributes both at decision nodes and leaves. Functional trees have been proposed and used by several researchers both in machine learning and statistics communities. Those works are oriented toward single algorithms, discussing different methods to generate the same kind of decision models. The proposed framework is oriented toward a family of algorithms, discussing the topological differences between decision models.

The most relevant contributions of this work are:

– A unified framework for various classification and regression functional trees.

– An in-depth study of the behavior of functional trees. Experimental results show that using functional leaves is a variance reduction method, while using functional inner nodes is a bias reduction process.
– New algorithms for classification and regression problems. To my knowledge, this is the first work that proposes the simultaneous use of functional nodes and functional leaves in prediction problems.
– The experimental study suggests that multivariate models using linear functions *both* at decision nodes and leaves exhibit some advantage, mostly in large datasets.

The paper is organized as follows. In the next section we present related work both in the classification and regression settings. In Section 3 we describe the framework used to analyze the behavior of functional inner nodes and functional leaves. In Section 4 we discuss the different variants of multivariate models using an illustrative example on regression domains. In Section 5 we evaluate the proposed algorithm on a set of benchmark regression and classification problems. The last section concludes the paper and presents directions for future research.

2. Related work

The standard algorithm to build univariate trees has two phases. In the first phase a large tree is constructed. In the second phase this tree is pruned back. The algorithm to grow the tree follows the standard divide-and-conquer approach. The most relevant aspects are: the splitting rule, the termination criterion, and the leaf assignment criterion. With respect to the last criterion, the usual rule is the assignment of a constant to a leaf node. Considering only the examples that fall at this node, the constant is usually the constant that minimizes the loss function: the mode of y values in the case of classification problems using the *0-1 loss function* or the mean of the y values in the regression setting using *mean-squared error*. With respect to the splitting rule, it is useful to distinguish between nominal attributes and continuous ones. In the latter the most common approaches uses a binary partition based on the value of a test in the form $att_i \leq cut\_point$. In the former the number of partitions is often equal to the number of values of the attribute although binary-subset-based splits ($att_i \in \{V_j, \ldots, V_k\}$) are also possible. To estimate the merit of the partition obtained by a given attribute several heuristics have been presented. A splitting rule typically works as a one-step lookahead heuristic. For each possible test, the system hypothetically considers the subsets of data obtained. The system chooses the test that maximizes (or minimizes) some heuristic function over the subsets. An example of the so-called *merit functions* is the *gain ratio* (Quinlan, 1993a) heuristic for classification problems and the *decrease in variance* (Breiman et al., 1984) criterion for regression problems. In any case, the attribute that maximizes the criterion is chosen as the test attribute at this node.

The pruning phase consists of traversing the tree in a depth-first order. At each non-leaf node two measures should be estimated. An estimate of the error of the subtree below this node, which is computed as a weighted sum of the estimated error for each leaf of the subtree, and the estimated error of the non-leaf node if it were pruned back to a leaf. If the latter is lower than the former, the entire subtree is replaced by a leaf.

All of these aspects have several important variants, see for example (Breiman et al., 1984; Quinlan, 1993a). Nevertheless all decision nodes contain conditions based on the values of one attribute, and leaf nodes predict a constant.

The CART book (Breiman et al., 1984) presents the first extensive and in-depth study of the problem of constructing decision trees for classification and regression problems. It is interesting to point out that while in the case of classification problems Breiman and colleagues consider internal nodes with a test based on linear combination of attributes, in the case of regression problems internal nodes are univariate. In the book (page 248), the authors write:

> "In regression, use of linear combinations splits does not have the appeal that it does in classification. (...) A promising alternative for improving accuracy is to grow a small tree using only a few of the most significant splits. Then do multiple linear regression in each of the terminal nodes."

We have done a set of experiments using the implementations of CART and Linear Regression in R (Ihaka & Gentleman, 1996), in the *housing* dataset to verify this observation. The linear regression built a model from the training set. Applying the linear model to the training and test set we obtain two vectors of predictions. The training and test sets are extended with the corresponding predictions. We compare the mean-squared error in the test set of CART when trained with the original dataset and with the extended dataset. Out of one hundred experiments, the insertion of the new attribute improved the performance of CART in 75 experiments. This is an indication that, in the regression setting, the use of linear combinations could improve decision trees. In the experimental section we will present more evidence of this fact.

In the context of classification problems, several algorithms have been presented that can use at each decision node tests based on a linear combination of the attributes: CART (Breiman et al., 1984), FACT (Loh & Vanichsetakul, 1988), OC1 (Murthy, Kasif, & Salzberg, 1994), LMDT (Utgoff & Brodley, 1991), LTREE (Gama, 1997), QUEST (Loh & Shih, 1997), and CRUISE (Kim & Loh, 2001). One of the most comprehensive studies of multivariate trees has been presented by Brodley and Utgoff (1995). In this work, the authors discuss several issues for constructing multivariate decision trees: representing a multivariate test, including symbolic and numeric features, learning the coefficients of a multivariate test, selecting the features to include in a test, and pruning of multivariate decision trees. However only multivariate tests at inner nodes in a tree are considered. A recent multivariate tree algorithm is the system Cruise. It was presented by Kim and Loh (2001). Cruise splits each node into as many sub-nodes as the number of classes in the target variable (as in FACT and LMDT); it uses analytical methods based on discriminant analysis to choose the splitting tests (instead of the exhaustive method used in C4.5, and Ltree), leading to negligible bias in variable selection. Quadratic discriminant analysis is used to find the coefficients of multivariate tests. We use Cruise in our experimental comparisons.

In the context of classification problems few works consider functional tree leaves. One of the earliest works is the *Perceptron Tree* algorithm (Utgoff, 1988) where leaf nodes may implement a general linear discriminant function. Kononenko, Cestnik, and Bratko (1988) and Kohavi (1996) have presented tree algorithms using functional leaves. These

hybrid algorithms generate a regular univariate decision tree, but the leaves contain a naive Bayes classifier built from the examples that fall at this node. The approach retains the interpretability of naive Bayes and decision trees, while resulting in classifiers that frequently outperform both constituents, especially in large datasets. A similar approach using bivariate *discriminant analysis* appears in Kim and Loh (2003). Recently Seewald, Petrak, and Widmer (2001) presented an interesting extension to this topology by allowing leaf nodes using different kind of models: naive Bayes and a multi-response linear regression, and instance-based models. The results indicate a certain performance improvement. Also, Gama and Brazdil (2000) have presented *Cascade Generalization*, a method to combine classification algorithms by means of constructive induction. The work presented here, follows closely the Cascade method but extended for regression domains and allowing models with functional leaves.

In regression domains, Quinlan (1992) presented the system M5. It builds multivariate trees using linear models at the leaves. In the pruning phase for each leaf a linear model is built. Recently Witten and Frank (2000) have presented M5′ a *rational reconstruction* of Quinlan's M5 algorithm. M5′ first constructs a regression tree by recursively splitting the instance space using tests on single attributes that maximally reduce variance in the target variable. After the tree has been grown, a linear multiple regression model is built for every inner node, using the data associated with that node and all the attributes that participate in tests in the subtree rooted at that node. Then the linear regression models are simplified by dropping attributes if this results in a lower expected error on future data (more specifically, if the decrease in the number of parameters outweighs the increase in the observed training error). After this has been done, every subtree is considered for pruning. Pruning occurs if the estimated error for the linear model at the root of a subtree is smaller or equal to the expected error for the subtree. After pruning terminates, M5′ applies a *smoothing* process that combines the model at a leaf with the models on the path to the root to form the final model that is placed at the leaf. Also Karalic (1992) studied the influence of using linear regression in the leaves of a regression tree. As in the work of Quinlan, Karalic shows that it leads to smaller models with increase of performance. Torgo (1997) has presented an experimental study about functional models for regression tree leaves. Later, the same author (Torgo, 2000) presented the system RT. RT is a system able to use several functional models at the leaves, including partial linear models. RT builds and prunes a regular univariate tree. Then at each leaf a linear model is built using the examples that fall at this leaf. In the regression setting few works consider multi-variable splits. One of them has been presented by Li, Lue, and Chen (2000) where decision nodes contain linear regressions and data is split according to the sign of the residuals.

In a more general context, multivariate and model trees are related to the problem of finding the appropriate bias for a given task. In this context those models could be related to *Stacked Generalization* (Wolpert, 1992) a technique used to combine algorithms with different representation languages. *Stacked Generalization* is a technique that uses learning at two or more levels. A learning algorithm is used to determine how the outputs of the base classifiers should be combined. Brodley (1995) presented the *Model Class Selection* (*MCS*) system, a hybrid algorithm that combines, in a single tree, nodes that are univariate tests, or multivariate tests generated by *linear machines* or instance-based learners. At each node

MCS uses a set of *if-then* rules to perform a heuristic best-first search for the best hypothesis for the given partition of the dataset. *MCS* uses a dynamic search control strategy, guided by a set of rules that incorporates knowledge of experts, to perform an automatic model selection. *MCS* builds trees which can apply a different model in different regions of the instance space. Recently Todorovski and Dzeroski (2003) have introduced meta decision trees, a method for combining classifiers. Meta decision trees are decision trees where leaves predict which classifier should be used to obtain a prediction. Meta decision trees are built from a meta dataset using properties of the class probability distributions of the predictions of base classifiers.

3. The framework for constructing functional trees

The algorithms reported in the previous section use functions at inner nodes or at leaves in decision trees. We denote these algorithms as functional trees. Those works are oriented toward single algorithms, discussing different methods to generate the same kind of decision models. In this section we present a framework that allow us to analyze and discuss *where* should we use combinations of attributes. The proposed framework applies to both classification and regression problems. It can be seen as an extension to multivariate and model trees.

3.1. Functional trees

Given a set of examples and an attribute constructor, the general algorithm used to build a functional tree is presented in figure 1. This algorithm is similar to many others, except in

Function GrowTree(Dataset, Constructor)

1. If Stop_Criterion(DataSet)
   - Return a Leaf Node with a constant value.
2. Construct a model Φ using Constructor
3. For each example $\vec{x} \in DataSet$
   - Compute $\hat{y} = \Phi(\vec{x})$
   - Extend $\vec{x}$ with new attributes $\hat{y}$.
4. Select the attribute of original as well as of newly constructed attributes that maximizes some merit-function
5. For each partition i of the DataSet using the selected attribute
   - $\text{Tree}_i = \text{GrowTree}(\text{Dataset}_i, \text{Constructor})$
6. Return a *Tree*, as a decision node based on the selected attribute, containing the Φ model, and descendants Tree_i.

End Function

Figure 1. Building a functional tree.

the constructive phase (steps 2 and 3). Here a function is built and mapped to new attributes. There are some aspects of this algorithm that should be made explicit. In step 2, a model is built using the constructor function. This is done using only the examples that fall at this node. Later, in step 3, the model is mapped to new attributes. The constructor function should be a classifier or a regressor depending on the type of the problem. In the former the number of new attributes is equal to the number of classes[1], in the latter the constructor function is mapped to one new attribute. In step 3, each new attribute is computed as the value predicted by the constructed function for each example. In the classification setting, each new attribute-value is the probability that the example belongs to one class given by the constructed model.

The merit of each new attribute is evaluated using the merit-function of the univariate tree, and in competition with the original attributes (step 4).[2] The model built by our algorithm has two types of decision nodes: those based on a test of one of the original attributes, and those based on the values of the constructor function. When using generalized linear models (GLM) as the attribute constructor, each new attribute is a linear combination of the original attributes. Decision nodes based on constructed attributes define a multivariate decision surface.

Once a tree has been built, it is pruned back. The general algorithm to prune the tree is presented in figure 2. The tree is traversed using a bottom-up, post-order strategy. For each non-leaf node two quantities are estimated: the *static error* and the *backed-up error*. *Static error* is an estimation of the error as if the node were a leaf. *Backed-up error* is the weighted sum of the estimation of the errors of all subtrees of the current node. The estimation of the error of each branch is weighted by the probability that an example follows the branch. If *backed-up error* is greater or equal than *static error*, then the node is replaced by a leaf with the majority class of the node.

The fundamental aspect of the pruning algorithm is the error estimation in step 1. At each node, we need to estimate the probability of error given the error in the sample of examples that fall at this node. The probability of error cannot be determined exactly. For a given *confidence level* we can obtain a confidence interval $[L_{cf}; U_{cf}]$ that with probability $1 - cf$ contains the true error. As in Quinlan (1993a), we use the upper limit of the confidence interval U_{cf} as a *pessimistic estimation* for the true error.

The confidence interval for the error depends on the loss-function used. In classification problems, and under the *0-1 loss*, the error is governed by a Binomial distribution (Mitchell, 1997). In regression problems, and for the *squared error loss* the variance of the error for a given example follows a χ^2 distribution. For a set of examples the sum of χ^2 can be approximated using a Normal distribution (Bhattacharyya & Johnson, 1977). For each node, the upper limit of the confidence interval is obtained from the sample error using the appropriate distribution. A similar procedure is used to estimate the constructor error (step 3).

The pruning algorithm produces two different type of leaves: *ordinary leaves* that predict a constant, and *constructor leaves* that predict the value of the constructor function learned (in the growing phase) at this node.

We obtain different conceptual models by simplifying our algorithm. Two interesting simplifications are described in the following sub-sections.

Function Prune(Tree)

1. Estimate **Leaf_Error** as the error at this node.
2. If Tree is a leaf Return **Leaf_Error**.
3. Estimate **Constructor_Error** as the estimated error of Φ.
4. For each descendent i
   - Let p_i the probability that an example goes through branch i.
   - **Backed_Up_Error** += $p_i \times Prune(Tree_i)$
5. If argmin(Leaf_Error,Constructor_Error,Backed_Up_Error)
   - Is Leaf_Error
     - Tree = Leaf
     - Tree_Error = Leaf_Error
   - Is Model_Error
     - Tree = Constructor Leaf
     - Tree_Error = Constructor_Error
   - Is Backed_Up_Error
     - Tree_Error = Backed_Up_Error
6. Return Tree_Error

End Function

Figure 2. Pruning a functional tree.

3.1.1. Functional leaves only. We denote as *FT-Leaves* the approach to functional trees where the functional models are not used in splitting-tests, but can be used at leaves. This is the strategy used for example in M5 (Quinlan, 1993b; Witten & Frank, 2000), and in the NBtree system (Kohavi, 1996). In our tree algorithm this is done by restricting the selection of the test attribute (step 4 in the growing algorithm) to the original attributes. Nevertheless we still build, at each node, the constructor function. The model built by the constructor function is used later in the pruning phase. In this way, all decision nodes are based on the original attributes. Leaf nodes could contain a constructor model. A leaf node contains a constructor model if and only if in the pruning algorithm the estimated error of the constructor model is lower than both the *Backed-up-error* and the *Static error*. A FT-Leaves functional tree divides the input space into hyper-rectangles. The data in each region could be fitted with the constructor function.

3.1.2. Functional inner nodes only. We denote as *FT-Inner* the approach to functional trees where the multivariate models are used exclusively at decision nodes (internal nodes) and not used as classifiers in leaves. In our algorithm, restricting the pruning algorithm to choose only between the *Backed-up-Error* and the *Static error* generates this kind of model. In this case all leaves predict a constant value. This is the strategy used in systems

like LMDT (Utgoff & Brodley, 1991), OC1 (Murthy, Kasif, & Salzberg, 1994), LTREE (Gama, 1997), and QUEST (Loh & Shih, 1997). A FT-Inner functional tree divides the input space into oblique decision surfaces. The data in each region is fitted with a constant that minimizes the given loss function.

3.1.3. Prediction using functional trees. Once a functional tree has been built, it can be used to predict the value of the target attribute for unclassified examples. As usual, the example traverses the tree from the root node to a leaf. At each decision node (inner node) of the tree the set of attributes of the example is extended using the constructor function built at this node. After this expansion, the decision test of the node is applied defining the path that the example will follow. When a leaf is reached the example is classified using either the constant associated with the leaf or the constructor function built at this leaf.

3.1.4. Other issues. A functional tree can use functional splitting tests and/or functional leaves. Although they play different roles in the system the algorithm that generates both is the same. Nevertheless, a functional splitting test is installed if and only if it minimizes the attribute evaluation function while a functional leaf is used if and only if it has lower expected error. Those functions act like intermediate concepts potentially useful for discriminating between classes. They are used for structuring the final model. The different uses of those models can provide different levels of detail.

As we have already pointed out, at different depths of the tree the number of attributes varies. To apply the attribute constructor, we can consider two variants. The attribute constructor can be applied using only the original attributes or using all the attributes at this level. In the latter, the function built at a given node will contain terms based on the terms built at previous nodes. To avoid this increase of complexity, in the experiments reported here, we restrict the constructor functions to the original attributes.[3]

Functional trees extend and generalize multivariate trees. Our algorithm can be seen as a hybrid model that performs a tight combination of a univariate tree and a GLM function. The components of the hybrid algorithm use different representation languages and search strategies. The tree uses a divide-and-conquer method; GLM functions perform a global minimization approach. The tree performs feature selection; GLM's use all (or almost all) the attributes to build a model. From the the bias-variance decomposition of the error (Breiman, 1998) point of view, a decision tree is known to have low bias but high variance, and GLM functions are known to have low variance but high bias. Different bias-variance profiles has been pointed out as a desirable property for components of hybrid models (Kohavi & Wolpert, 1996).

4. An illustrative example

In this section we use the well-known regression dataset *Housing* to illustrate the different variants of functional models. The attribute constructor used is the linear regression function. Figure 3(a) presents a univariate tree for the *Housing* dataset. Decision nodes contain only tests based on the original attributes. Leaf nodes predict the average of y values taken from the examples that fall at the leaf.

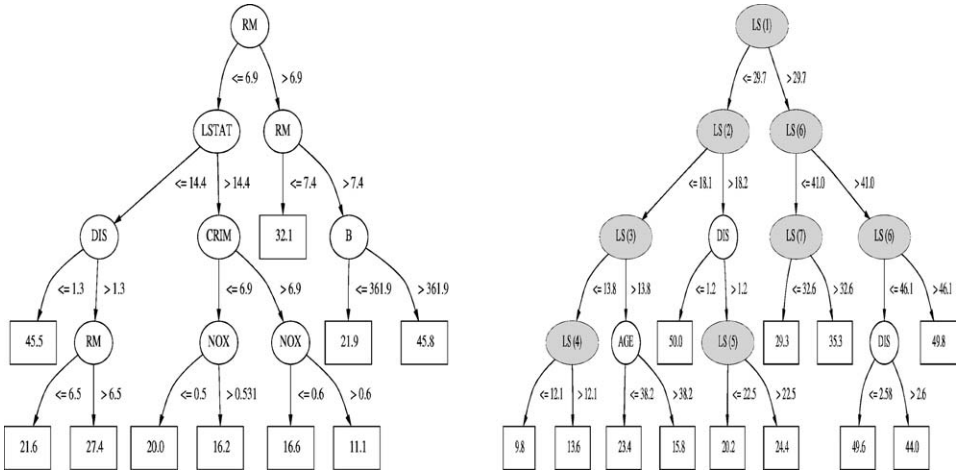

Figure 3. (a) The univariate regression tree and (b) FT-Inner regression tree for the housing problem.

In a FT-Inner multivariate tree (figure 3(b)) decision nodes can contain tests based on a linear combination of the original attributes. The tree contains a mixture of nodes with tests based on learned attributes, denoted as *LS*, and original attributes, e.g. *AGE, DIS*. Any of the linear-regression attributes can be used at the node where they have been created and at deeper nodes. For example, the *LS* (6) has been created at the second level of the tree. It is used as the test attribute at this node, and also (due to the constructive ability) as the test attribute at the third level of the tree. Leaf nodes predict the average of y values of the examples that fall at this leaf. In a FT-Leaves multivariate tree (figure 4(a)) the

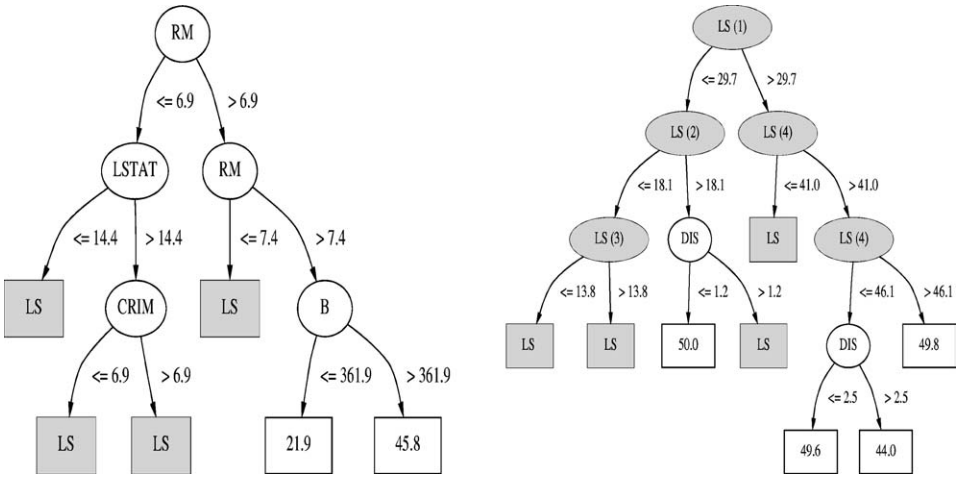

Figure 4. (a) The FT-Leaves functional regression tree and (b) The full functional regression tree for the housing problem.

decision nodes only contain tests based on the original attributes. Leaf nodes could predict values using a linear-regression function. In such cases, the function is built only using the examples that fall at this node. This is the type of multivariate regression tree that usually appears in the literature. For example, systems M5 (Quinlan, 1993b; Witten & Frank, 2000) and RT (Torgo, 2000) generate this type of models. Figure 4(b) presents the full multivariate regression tree using both functional inner nodes and functional leaves. In this case, decision nodes can contain tests based on a linear combination of the original attributes, and leaf nodes can predict the values obtained by using a linear-regression function built from the examples that fall at this node.

In the set of experiments described in the next section using the *Housing* problem, the mean square error of the univariate tree was 19.59, FT-Leaves 13.36, and FT-Inner 16.54 and FT 16.25.

5. Experimental evaluation

The main goal in this experimental evaluation is to study the influence, in terms of performance, of the use of linear models inside regression and classification trees: How do the algorithms derived from this framework compare against the state of the art in classification and regression trees? What is the impact in the performance of the final model of functional leaves and functional nodes? What is the role of each component?

We evaluate the full model implemented in the framework, its simplified variants, and its components on a set of classification and regression benchmark problems. We evaluate three situations:

– Trees that can use linear combinations at each internal node.
– Trees that can use linear combinations at each leaf.
– Trees that can use linear combinations both at internal nodes and leaves.

Using the common framework allows all evaluated models to be based on the same tree growing and pruning algorithm. That is, they use exactly the same splitting criteria, stopping criteria, and pruning mechanism. In regression problems the constructor is a standard linear regression function. In classification problems the constructor is the LinearBayes classifier (Gama, 2000).[4] Moreover they share many minor heuristics that individually are too small to mention, but collectively can make a difference. Doing so, the differences on the evaluation statistics are solely due to the differences in the conceptual model.

5.1. Experimental setup

We estimate the performance of a learned model using 10-fold cross validation. To minimize the influence of the training set variability, we repeat this process ten times, each time using a different permutation of the examples in the dataset. The final estimate is the mean of the performance statistics obtained in all runs of the cross validation. For regression problems the performance is measured in terms of *mean squared error*. For classification problems the performance is measured in terms of *error rate*. To apply pairwise comparisons we

guarantee that, in all runs, all algorithms learn and test on the same partitions of the data. For each dataset, comparisons between algorithms are done using the *Wilcoxon signed ranked paired-test*. The null hypothesis is that the difference between performance statistics has median value zero. We consider that a difference in performance has statistical significance if the *p value* of the Wilcoxon test is less than 0.05 after a Bonferroni multiple comparisons correction.

5.1.1. Organization of the experiments. The experimental evaluation of the different algorithms has been organized into four main dimensions. The goal of the two first sets of experiments is to provide evidence that the proposed framework can implement algorithms that are competitive with the state of the art in univariate, multivariate and model trees. Based on these results we argue that the framework is appropriate to discuss the role of linear combinations in decision nodes and leaves in the performance of the final model (last two sets of experiments). The goals of the experiments are as follows:

- The simplest model that can be derived from the framework is a univariate tree. Is the baseline of the framework competitive against the state of the art in classification and regression univariate trees? The first set of experiments answers this question. We compare the univariate tree against well known classification and regression univariate trees.
- How does the functional tree (FT) compare against the state of the art in multivariate classification and regression trees? In a second set of experiments, we compare the performance of the functional tree against the state of the art in multivariate classification and regression trees.
- Functional trees generate hybrid models by tightly coupling two other classifiers. Is the integration method of the different models sound? In a third set of experiments, we compare the performance of the functional tree (FT) against its components: the univariate tree (UT) and the constructor function, linear regression (LR) in regression problems, and LinearBayes (LB) in classification problems.
- What is the role of functional leaves and functional nodes? Do they have different influence in the performance of the final model? This is the main problem we discuss in the paper. To answer this question we compare the functional tree against the two simplified versions: FT-Leaves and FT-Inner.

For comparative purposes we present the results of other well-known algorithms. For regression problems we have used system M5′[5] with and without smoothing, and two versions of system RT (version 4.1.1): univariate and *partial linear trees*. For classification problems, we have used system C4.5 (release 8.0), system M5′Class (Frank et al., 1998), and two versions of system Cruise (version 1.11): univariate and multivariate trees.

5.2. Results for classification problems

We have chosen 30 datasets from the UCI repository (Blake, Keogh, & Merz, 1999). All of them them have been previously used in comparative studies. The number of examples

varies from 150 to 52202. The number of attributes varies from 4 to 60, and the number of classes from 2 to 26. For comparative purposes we also evaluated the state of the art in univariate trees, C4.5 (Quinlan, 1993a), M5′Class (Witten & Frank, 2000), and the univariate and multivariate versions of Cruise (Kim & Loh, 2001). C4.5 and univariate Cruise use univariate splits and constants at leaves, M5′Class uses univariate splits and linear models at leaves, multivariate Cruise uses linear combinations at inner nodes and constants at leaves.

The detailed results in terms of error-rate are presented in the Appendix in Tables A-1, A-2, and A-3. The results of LinearBayes and univariate trees are presented in Table A-1. The reference algorithm is the univariate tree of our framework. In Table A-2 we present the results of multivariate trees. The algorithm used for comparison is the functional tree (FT). In Table A-3 we present the results of functional trees and their components. The reference algorithm is again the functional tree (FT). For each dataset, the algorithms are compared against the reference algorithm using the *Wilcoxon signed ranked paired-test*. A $-$ $(+)$ sign indicates that for this dataset the performance of the algorithm was worse (better) than the reference model with a *p value* less than 0.05 after Bonferroni correction.

Those tables are summarized in Tables 1, 2, and 3. All summary tables are organized as follows. The first two lines present the arithmetic and the geometric mean of the error rate across all datasets. The third line shows the average rank of all models, computed for each dataset by assigning rank 1 to the best algorithm, 2 to the second best and so on. The fourth line shows the average ratio of error rates. This is computed for each dataset as the ratio between the error rate of each algorithm and the error rate of the reference algorithm. The next two lines show the number of Wins/Losses and the number of significant differences using the *signed-rank test* with respect to the reference algorithm. We use the *Wilcoxon signed ranked paired-test* to compare the error rate of pairs of algorithms across datasets.[6] The last line shows the *p values* associated with this test for the results on all datasets with respect to the reference algorithm.

Table 1 presents a comparative summary of the results of simple algorithms: all the univariate trees and LinearBayes. In this case the reference algorithm is FT-Univ. All the statistics provide evidence that the univariate tree that is the basis of our framework is

Table 1. Summary of results of error-rate in classification problems of univariate tree and Linear-Bayes. The reference algorithm is the univariate tree.

	FT-Univ	Linear-Bayes	C4.5	Cruise-Univ.
Arithmetic mean	14.58	15.31	14.51	16.35
Geometric mean	9.741	11.632	9.399	8.897
Average rank	2.267	2.367	2.300	3.067
Average error ratio	1	8.2617	0.9813	1.6481
Wins (for FT-Univ)/losses	–	15/15	14/14	22/8
Significant wins/losses	–	14/13	5/3	17/4
Wilcoxon test	–	0.7266	0.7241	0.0066

Table 2. Summary of results of error-rate in classification problems of multivariate trees. The reference algorithm is FT.

	FT	M5′Cl	Cruise
Arithmetic mean	11.72	12.77	12.93
Geometric mean	6.62	7.24	6.69
Average rank	1.83	2.20	1.97
Average error ratio	1.0	1.23	2.09
Wins (for FT)/losses	–	18/12	17/13
Significant wins/losses	–	4/2	10/10
Wilcoxon test	–	0.13	0.39

comparable in generalization performance to C4.5, the state of the art in univariate trees. It is very different from LinearBayes. In 30 datasets there are 27 significant differences!

Table 2 presents a comparative summary of the results of the state of the art in multivariate trees. The reference algorithm is the full model functional tree (FT). All the evaluation statistics show that FT is a competitive algorithm. It is the algorithm with lowest average rank, arithmetic and geometric means. FT performs significantly better than M5′ on four datasets. We should note that M5′ decomposes a n-classes classification problem into $n - 1$ binary regression problems. With respect to Cruise there is a draw with respect to the significant wins. FT significantly outperforms Cruise on the larger datasets (Adult, Letter, Satimage, Shuttle).

Table 3 presents a comparative summary of the results of all classification algorithms implemented in our framework: the components and variants of functional trees. The full model (FT) is the algorithm with lowest average rank, average error ratio, arithmetic and geometric means. FT significantly improves over both components (LB and FT-Univ) in 6 datasets. With respect to its components, the p-values of the Wilcoxon Test indicates that the performance of FT is significantly better than each component at a significance level greater than 99%. All the multivariate trees have a similar performance. Nevertheless, in all statistics, FT is the algorithm with best performance. It is interesting to note also

Table 3. Summary of results of error-rate in classification problems. Comparison between components of functional trees and functional trees variants. The reference algorithm is FT.

	FT	FT-Leaves	FT-Inner	FT-Univ	LBayes
Arithmetic mean	11.72	12.56	11.90	14.58	15.31
Geometric mean	6.62	6.97	6.80	9.03	11.63
Average rank	2.567	2.617	2.833	3.483	3.5
Average error ratio	1.0	1.11	1.03	1.41	7.55
Wins (for FT)/losses	–	15/13	11/7	22/8	20/10
Significant wins/losses	–	8/7	3/0	14/5	15/5
Wilcoxon test	–	0.724	0.055	0.0015	0.0015

that the FT-Inner version is the most competitive algorithm. A more detailed analysis of the comparison between FT and its simplified versions (FT-Inner and FT-Leaves) will be presented later in Section 5.4.

5.3. Results in regression domains

We have chosen 20 datasets from the *Repository of Regression problems at LIACC*.[7] The choice of datasets was restricted using the criterion that almost all the attributes be ordered with few missing values.[8] The number of examples varies from 43 to 40768. The number of attributes varies from 5 to 48.

The detailed results in terms of *mean square error* (MSE) are presented in Tables A-4, A-5, and A-6. In Table A-4 we present the results of Linear Regression and univariate trees. The algorithm used for comparison is the univariate tree of our framework. In Table A-5 we present the results of model trees. The algorithm used for comparison is the functional tree of our framework. In Table A-6 we present the results of functional trees, and the simplified versions of functional trees. The algorithm used for comparison is the full functional tree. In all tables, and for each dataset, all the algorithms are compared against the algorithm taken as reference using the *Wilcoxon signed rank-test*. A $-$ $(+)$ sign indicates that in this dataset the performance of the algorithm was worse (better) than the reference algorithm with a *p value* less than 0.05 after Bonferroni correction.

Tables 4, 5, and 6 present summaries of the results according to the methodology presented in Section 5.1.1. In each table we use as reference the most appropriate algorithm for the comparative analysis: FT-Univ for Table 4, and FT for Tables 5 and 6. All tables are organized similarly to the summary tables for classification.

Table 4 presents a comparative summary of the results of simpler algorithms: the univariate trees (FT-U and RT-U) and Linear Regression. In this case the reference algorithm is FT-Univ. RT-Univ is the algorithm with the best performance. A possible reason for that is the cost-complexity pruning used by RT. All the statistics provide evidence that the univariate tree, which is the basis of our framework, has a performance similar to the state of the art in univariate trees.

Table 5 presents a comparative summary of the results of the state of the art in model trees. The reference algorithm is the full model functional tree (FT). FT exhibits a performance

Table 4. Summary of results of mean-squared error in regression problems of univariate trees and linear regression.

	FT-Univ	LR	RT-Univ
Geometric mean	23.59	39.17	22.22
Average rank	1.9	2.4	1.7
Average Error Ratio	1.0	2.94	0.96
Wins (for FT-Univ)/losses	–	13/7	9/11
Significant wins/losses	–	13/5	3/8
Wilcoxon test	–	0.105	0.38

Table 5. Summary of results of mean-squared error of model trees in regression problems.

	FT	RT-PLS	M5′	M5′nSmooth
Geometric mean	16.90	20.04	16.09	17.76
Average rank	2.2	3.1	1.8	2.9
Average error ratio	1.0	1.37	0.98	1.07
Wins (for FT)/losses	–	17/3	9/11	11/8
Significant wins/losses	–	12/1	2/3	4/3
Wilcoxon test	–	0.0012	0.96	0.1776

Table 6. Summary of results of mean-square error in regression problems. Comparison between components of functional trees and functional trees variants.

	FT	FT-Leaves	FT-Inner	FT-Univ	LR
Geometric mean	16.90	16.48	17.68	23.59	39.17
Average rank	2.025	2.275	2.400	3.900	4.400
Average error ratio	1.00	0.99	1.07	1.53	3.4973
Wins (for FT)/losses	–	11/6	12/8	16/4	19/1
Significant wins/losses	–	6/4	7/6	15/3	18/0
Wilcoxon test	–	0.67	0.648	0.003	0.000

similar to M5′, the best performing algorithm. All the evaluation statistics show that FT is a competitive algorithm. It is interesting to observe the influence of smoothing in the behavior of M5′. In terms of significant wins, M5′ significantly outperforms the version without smoothing in 8 datasets and never is worst. Smoothing takes into account the linear models built at inner nodes in the tree. This result is relevant to the conclusions detailed below.

Table 6 presents a comparative summary of the results of all regression algorithms implemented in our framework: the components and variants of functional trees. The full model (FT) significantly improves over both components (LR and FT-U) in 14 datasets out of 20. All the multivariate trees have a similar performance. Considering only the number of significant wins, FT exhibits some advantage. The FT-Leaves version is the most competitive algorithm. The ratio of significant wins/losses between the FT-Leaves and FT-Inner versions is 4/3. Nevertheless there is a computational cost associated with the increase in performance. To run all the experiments referred to here, FT requires almost 1.8 more time than the univariate regression tree.

5.4. Discussion

In this subsection we discuss in detail the performance of functional tree variants. The main goal of this analysis is to understand the role and the impact in the performance of the final model of functional nodes and functional leaves.

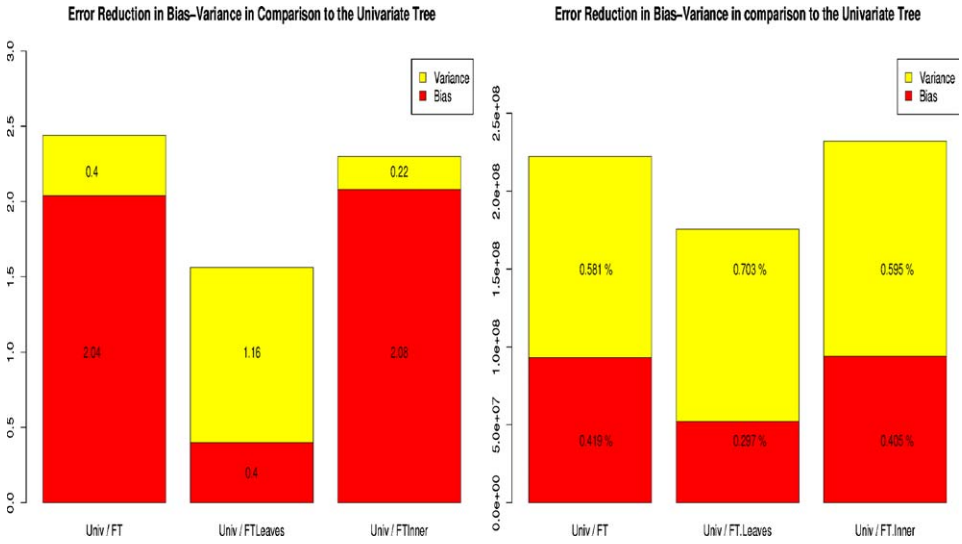

Figure 5. Reduction in bias and variance of functional trees variants with respect to a univariate tree: (left) Classification problems and (right) Regression problems.

Several authors state that there is a trade-off between the systematic errors due to the representational language used by an algorithm (the *bias*) and the *variance* due to the dependence of the model to the training set (Breiman, 1998). When using more flexible representations the bias is reduced. We have tested this hypothesis. We have compared the bias-variance error decomposition of the univariate tree against the functional trees. For classification problems we have used the decomposition suggested by Kohavi and Wolpert (1996), while for regression problems we have used the standard decomposition presented in Geman, Bienenstock, and Doursat (1992). Figure 5 shows the reduction in the error components of each functional tree with respect to the univariate tree. In classification problems, 90% of the error reduction is due to the bias component. In regression problems, most of the reduction is due to a reduction in variance. FT is the algorithm with lowest average bias. FT-Leaves is the algorithm with lowest average variance. In classification, FT and FT-Inner have similar bias and variance reductions. FT-Leaves shows a different behavior: the main error-reduction is due to a reduction in variance. Although not so clear as in classification, a similar observation applies to regression. It is known that discriminant analysis is stable to perturbations of the training set. In the case of FT-Leaves, deeper nodes in the tree (nodes where decisions are based on small number of examples) are replaced by stable classifiers, leading to a decrease in variance. FT-Inner uses splits based on combinations of attributes, introducing more powerful and flexible representations, leading to a reduction in bias. These results suggest that standard variance reduction methods, like *bagging*, will be more effective with FT and FT-Inner than with FT-Leaves.

5.4.1. Bagging functional trees. One of the most attractive features of a univariate tree is that we can reduce its error by applying simple variance reduction methods. Does this feature

apply to a multivariate tree? We have applied a standard variance reduction method, *bagging* (bootstrap aggregating (Breiman, 1996)), to our functional tree (FT). In our experiments we have used bagging by aggregating 10 trees. The results report the mean of 10-fold cross validation. Detailed results are reported in Tables A-7 and A-8. On both types of problem the results are similar. On these datasets, bagging FT improves the overall performance of FT. Bagging significantly improves the performance on 7 regression problems and 3 classification datasets. Moreover, on these datasets we never observe a significant degradation of the performance. These results suggest that variance reduction methods can be combined with bias-reduction methods. This deserves future study. Moreover, in comparison to FT-Leaves, bagging FT-Leaves significantly improves in 2 classification problems and degrades the performance in 1 dataset (Monks-1), while in regression, bagging FT-Leaves significantly improves in 5 datasets. The use of functional leaves is also a variance reduction method, which explains the reduced impact of bagging in FT-Leaves.

5.4.2. The effect of training-set size. Several authors have pointed out that the increase in the size of datasets naturally will reduce the variance of tree models (Brain & Webb, 2002). In that paper the authors wrote: "*Experiments show that learning from large datasets may be more effective when using an algorithm that places greater emphasis on bias management*". This is confirmed in the experiments described in this paper, where FT-Inner is significantly better than FT-Leaves in the largest datasets: Adult, Letter, Satimage.

To study the influence of training-set size on the behavior of functional trees, we have done a set of learning-curve experiments using the *Adult* (classification) and *Friedman* (regression) datasets. Experiments were performed as follows. We fixed a hold-out test set with 4000 examples. From the remainder, we obtain training sets with increasing numbers of examples. For each training set we measure the test-set error, the bias and the variance components of the error. The algorithms used were FT-Inner and FT-Leaves. For both algorithms the error, the bias, and the variance slowly decrease when increasing the number of examples. In figure 6 we plot the differences (FT-Leaves–FT-Inner). We observe that while the differences in bias are always positive and increase with training-set size, the differences in variance are negative and decrease with training-set size. These results confirm

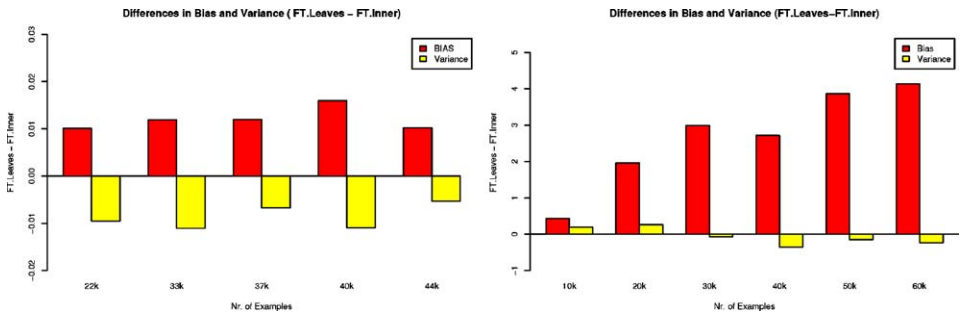

Figure 6. Differences in bias and variance between FT-Leaves and FT-Inner varying the number of training examples in *Adult* classification dataset (left) and *Friedman* regression dataset (right).

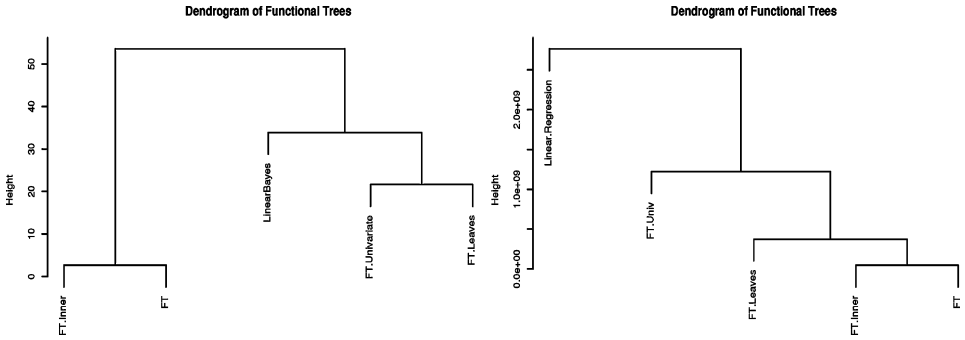

Figure 7. Dendrogram of similarities between error-rates of functional trees for classification problems (left) and regression problems (right).

the different impact in the components of the error of functional inner nodes and functional leaves. Functional inner nodes is a technique for bias management, and functional leaves emphasis in variance management.

5.4.3. Similarities between results of functional trees. We have used a hierarchical clustering algorithm to study the similarities between the errors of algorithms. In this analysis the similarity between two algorithms is given by the sum of differences between the errors of both algorithms in the *same* dataset. For each type of problem, the data matrix for cluster analysis has been constructed as follows. Each column represents one algorithm. Each row represents one dataset. The value of each cell is the error statistic of a given algorithm on the corresponding dataset.

Figure 7 shows the dendrogram obtained by applying cluster analysis respectively to the results of classification and regression algorithms of our framework. On both types of problem, the cluster {FT, FT-Inner} corresponds to the minimum distance level—that is maximum similarity. The similarity between FT and FT-Leaves is at least one order magnitude lower. This shows that the highest correlation between errors correspond to the clusters {FT, FT-Inner}. It provides additional evidence that FT and FT-Inner have very similar behavior.

Figure 8 shows the dendrograms obtained by applying cluster analysis to the results of all classification and regression algorithms used in our experimental study. For both types of problems there are three well-defined clusters. A cluster with algorithms using univariate decision nodes, a cluster with algorithms using multivariate decision nodes, and a cluster with algorithms using discriminant analysis. In classification problems some key clusters are: the univariate models {FT.Univ, C4.5}, functional leaves: {FT-Leaves, M5Class}, and functional nodes {FT-Inner, FT}. Interestingly, LinearBayes and Cruise appear in the same cluster. Both are based on discriminant analysis. In regression problems, M5′ without smoothing appears together with the other univariate trees, while M5′ appears together with FT-Inner and FT.

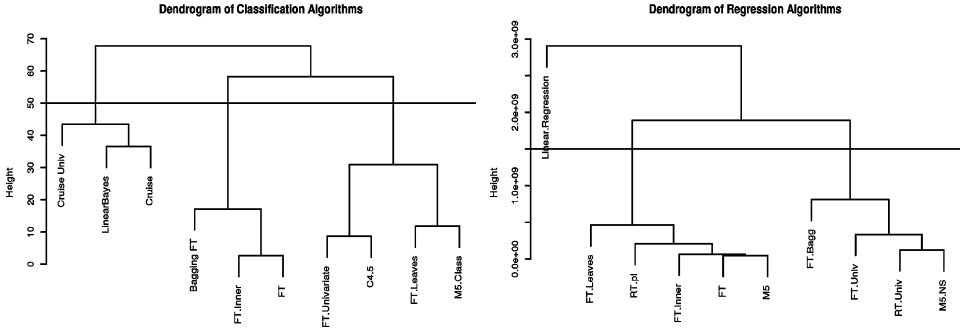

Figure 8. Dendrogram of similarities between error-rates of all used models for classification problems (left) and regression problems (right).

5.4.4. Model complexity. In this work we estimate the complexity of tree models by measuring the number of leaves. This measures the number of different regions into which the instance space is partitioned. On these datasets, the average number of leaves for the univariate tree is 81. All functional tree generate smaller models. The average number of leaves of the full model is 50, for FT-Leaves is 56, and for FT-Inner is 52.

The algorithms implemented in the framework have different degrees of complexity. Univariate trees, using tests based on a single attribute and constants in leaves, generate simpler but large models. By introducing functional nodes or functional leaves the level of complexity increases, but the global size of the model decreases. Using functional nodes and functional leaves corresponds to the most complex and smallest models.

5.4.5. Learning times. We have mentioned that there is a computational cost associated with the increases in performance observed. To run all the experiments discussed here, the fastest algorithm was the univariate tree and the slowest was FT. FT requires almost 1.8 more time than the univariate tree. FT-Leaves is the fastest functional tree. This is an indication that the most time-consuming operation in functional trees is not the construction of new attributes but the time needed to evaluate them. The new attributes are continuous and require a sort operation to estimate the merit of the attribute.

6. Conclusions

In this work we have presented functional trees, a framework to construct multivariate trees for regression and classification problems. From the framework it is possible to derive algorithms able to use functional decision nodes and functional leaf nodes. Functional decision nodes are built when growing the tree, while functional leaves are built when pruning the tree. A contribution of this work is to provide a single framework for various classification and regression multivariate trees. Functional trees can be seen as a generalization of prior multivariate trees for decision problems and model-trees for regression problems.

In the experimental evaluation on a set of benchmark problems we have compared the performance of the full functional tree against its components, two simplified versions, and the

state of the art in multivariate trees. The experimental evaluation points out some interesting observations. All multivariate tree versions have similar performance. On these datasets, there is no clear winner between the different versions of functional trees. Any functional tree out-performs its constituents (the univariate tree and the discriminant function) in a large set of problems.

In our study the results are consistent on both types of problem. For classification and regression we obtain similar rankings of the performance between the different versions of the functional algorithms. The experimental study suggests that the full model, which is a multivariate model using linear functions *both* at decision nodes and leaves, exhibits a marginal advantage. Although most of the prior work in multivariate classification trees follows the FT-Inner approach, the FT-Leaves approach seems to be an interesting and competitive alternative. A similar observation applies to regression problems. The FT-Inner approach deserves more attention for future research.

The analysis of the bias-variance decomposition of the error indicates that the use of multivariate decision nodes is a bias reduction process while the use of multivariate leaves is a variance reduction process. This analysis is a contribution to our understanding of how and why learning algorithms work. Methods like bagging, are very effective in variance reduction for decision trees (Breiman, 1998). Nevertheless the increase in the size of datasets that we verify nowadays, naturally will reduce the variance of tree models (Brain & Webb, 2002). This fact will increase the research on bias-reduction methods.

6.1. Contributions

Functional trees have been proposed and used by several researchers both in machine learning and statistics communities. Prior work has been oriented toward single algorithms, discussing different methods to generate the same kind of decision models. The framework introduced here is oriented toward a family of algorithms, discussing the topological differences between decision models. The most important contributions of this work are:

- A unified framework for various types of classification and regression functional trees.
- An in-depth study of the behavior of functional trees. The understanding of the behavior of algorithms is a fundamental aspect of machine learning research. The patterns that emerge from the bias-variance decomposition of the error, and the cluster analysis of similarities between functional trees provide strong insights to the behavior of those algorithms: variance can be reduced using functional leaves, while bias can be reduced using functional inner nodes.
- New algorithms for classification and regression problems. To my knowledge, this is the first work that proposes the simultaneous use of functional nodes and functional leaves in prediction problems. The topologically different models, that is functions in leaves and/or inner nodes, can provide different levels of interpretability for the problem under study.
- The experimental study suggests that all multivariate models are competitive. Nevertheless, using linear functions *both* at decision nodes and leaves exhibits some advantage mostly in large datasets. This observation can open new directions for future research.

6.2. *Limitations and future directions*

Univariate trees are invariant to monotonic transformations of the input variables, are robust to the presence of outliers, redundant and irrelevant attributes. Decision trees using combinations of attributes either in the splitting test in decision nodes or models at leaves could lose these properties. The application of the ideas we have presented in this paper to real-world contexts will require effective methods for feature selection in the attribute constructor.

Functional trees allow a tight integration of different representation languages for generalizing examples. It is interesting to note that there are standard algorithms for topological transformations on decision trees. For example, Sahami has presented an algorithm to transform a multivariate decision tree into a three-layer feed forward neural net (Sahami, 1995). Sahami argues that the method could be used to simultaneously learn the topology and the parameters of a neural network. Also Quinlan (1993a), and Frank and Witten (1998) have presented algorithms to transform a decision tree into a set of rules. Similar algorithms could be applied to functional trees. This is an interesting aspect because it points out a common representation for different representations languages. We intend to explore these aspects in future research.

Appendix

Table A-1. Error rates of LinearBayes and univariate trees in classification problems. For the comparisons at the bottom, the reference algorithm is FT-Univariate.

Dataset	C4.5	Cruise-Univ	LinBayes	FT Univ.
Adult	13.98 ± 0.56	-21.53 ± 0.81	-17.01 ± 0.49	14.18 ± 0.46
Australian	14.43 ± 0.64	14.32 ± 0.66	$+13.50 \pm 0.27$	14.75 ± 0.96
Balance	21.93 ± 0.7	-31.61 ± 1.8	$+13.35 \pm 0.35$	22.47 ± 1.1
Banding	23.60 ± 1.3	-26.31 ± 0.89	23.68 ± 0.97	23.51 ± 1.8
Breast(W)	-5.396 ± 0.33	5.148 ± 0.39	$+2.862 \pm 0.093$	5.123 ± 0.19
Cleveland	21.23 ± 2	-22.7 ± 0.75	$+16.13 \pm 0.39$	20.99 ± 1.4
Credit	14.51 ± 0.67	14.82 ± 0.36	14.23 ± 0.15	14.61 ± 0.55
Diabetes	25.28 ± 1.2	25.24 ± 0.55	$+22.71 \pm 0.21$	25.35 ± 1
German	28.29 ± 1.0	$+25.50 \pm 0.92$	$+24.52 \pm 0.23$	28.24 ± 0.73
Glass	33.54 ± 2.5	-39.86 ± 3.5	-36.65 ± 0.8	32.15 ± 2.3
Heart	22.07 ± 1.5	22.86 ± 0.67	$+17.70 \pm 0.23$	23.07 ± 1.7
Hepatitis	-21.55 ± 2	-24.84 ± 1.3	$+15.48 \pm 0.71$	17.13 ± 1.3
Ionosphere	10.09 ± 1.1	10.24 ± 0.81	-13.38 ± 0.76	10.03 ± 0.93
Iris	-4.933 ± 0.78	4.812 ± 0.42	$+2 \pm 0$	4.333 ± 0.79
Letter	11.64 ± 0.46	-36.12 ± 1.6	-29.82 ± 1.3	11.88 ± 0.56
Monks-1	$+3.522 \pm 1.6$	-13.87 ± 2.0	-25.01 ± 0.030	10.54 ± 1.7
Monks-2	32.86 ± 0	$+5.094 \pm 1.5$	-34.19 ± 0.58	32.86 ± 0

(*Continued on next page.*)

Table A-1. (*Continued*).

Dataset	C4.5	Cruise-Univ	LinBayes	FT Univ.
Monks-3	1.388 ± 0.01	$+1.356 \pm 0.009$	-4.16 ± 0.01	1.572 ± 0.39
Mushroom	0 ± 0	0.0025 ± 0.008	-3.11 ± 0.037	0 ± 0
Optdigits	9.48 ± 0.32	-16.85 ± 0.33	$+4.687 \pm 0.059$	9.476 ± 0.27
Pendigits	3.491 ± 0.13	-6.116 ± 0.16	-12.43 ± 0.036	3.559 ± 0.14
Pyrimidines	$+5.307 \pm 0.17$	$+5.061 \pm 0.16$	-9.846 ± 0.11	5.733 ± 0.24
Satimage	-13.66 ± 0.4	-15.6 ± 0	-16.01 ± 0.13	12.89 ± 0.23
Segment	3.221 ± 0.22	-4.25 ± 0.33	-8.407 ± 0.09	3.381 ± 0.21
Shuttle	0.028 ± 0.015	-0.39 ± 0.11	-5.63 ± 0.3	0.028 ± 0.025
Sonar	27.47 ± 2.3	25.55 ± 1.7	24.95 ± 1.2	27.65 ± 3.5
Vehicle	27.37 ± 1.2	-30.6 ± 0.74	$+22.16 \pm 0.15$	27.33 ± 1.2
Votes	$+3.355 \pm 0.28$	-4.313 ± 0.14	-9.74 ± 0.21	3.773 ± 0.47
Waveform	-24.92 ± 0.8	-26.36 ± 0	$+14.94 \pm 0.22$	24.04 ± 0.8
Wine	6.724 ± 1.2	-9.26 ± 1.8	$+1.133 \pm 0.45$	6.61 ± 1.3
Average mean	14.51	16.35	15.31	14.58
Geometric mean	9.399	8.897	11.632	9.741
Average ranks	2.300	3.067	2.367	2.267
Average error ratio	0.9813	1.6481	8.2617	1
Wins/losses	15/15	8/22	14/14	–
Significant wins/losses	3/5	4/17	13/14	–
Wilcoxon test	0.7241	0.0066	0.7266	–

Table A-2. Error rates of multivariate trees in classification problems. For the comparisons at the bottom, the reference algorithm is FT.

Dataset	Cruise	FT	M5′Class
Adult	-21.53 ± 0.81	13.83 ± 0.38	-15.18 ± 0.62
Australian	13.97 ± 0.19	13.64 ± 0.58	14.64 ± 5.2
Balance	8.22 ± 0.073	7.555 ± 0.81	-13.89 ± 3.2
Banding	-28.06 ± 0.9	23.93 ± 2	22.62 ± 5.3
Breast(W)	3.666 ± 0.24	3.346 ± 0.39	5.137 ± 3.1
Cleveland	$+15.86 \pm 0.32$	16.67 ± 0.78	17.93 ± 8
Credit	$+14.07 \pm 0.27$	15.22 ± 0.55	14.91 ± 3.7
Diabetes	23.67 ± 0.34	23.57 ± 0.82	25 ± 4.8
German	-27.41 ± 0.35	24.4 ± 0.72	26.3 ± 3.1
Glass	-39.27 ± 1.4	35.18 ± 2.4	29.48 ± 10
Heart	$+15.71 \pm 0.56$	17.19 ± 0.8	16.67 ± 9
Hepatitis	16.24 ± 1.1	16.67 ± 1.3	19.92 ± 8.5

(*Continued on next page.*)

Table A-2. (*Continued*).

Dataset	Cruise	FT	M5′Class
Ionosphere	$+9.229 \pm 0.83$	11.04 ± 1.6	9.704 ± 4.1
Iris	1.938 ± 0.20	2.06 ± 0.21	5.333 ± 5.3
Letter	-36.12 ± 1.6	11.83 ± 1.1	$+9.44 \pm 0.46$
Monks-1	$+4.267 \pm 1.6$	9.023 ± 2.0	10.05 ± 9
Monks-2	11 ± 2.6	9.213 ± 1.8	27.66 ± 21
Monks-3	$+1.356 \pm 0.0092$	2.929 ± 0.52	1.364 ± 2.4
Mushroom	$+0.0037 \pm 0.008$	0.111 ± 0.04	0.0246 ± 0.08
Optdigits	$+3.012 \pm 0.1$	3.304 ± 0.13	-5.429 ± 1.4
Pendigits	$+1.532 \pm 0.064$	2.892 ± 0.086	$+2.419 \pm 0.37$
Pyrimidines	-7.405 ± 0.20	6.158 ± 0.17	6.175 ± 0.95
Satimage	-13.89 ± 0.11	11.78 ± 0.32	12.40 ± 3.2
Segment	-3.806 ± 0.23	3.208 ± 0.21	2.468 ± 0.76
Shuttle	-1.177 ± 0.26	0.036 ± 0.025	0.067 ± 0.03
Sonar	23.79 ± 1.5	24.83 ± 2.7	22.72 ± 9
Vehicle	20.98 ± 0.46	20.97 ± 1.2	20.9 ± 4.6
Votes	-4.269 ± 0.010	3.703 ± 0.44	4.172 ± 4.0
Waveform	$+14.81 \pm 0.18$	15.91 ± 0.44	-17.24 ± 1.4
Wine	1.540 ± 0.47	1.404 ± 0.29	3.830 ± 3.6
Average mean	12.93	11.72	12.77
Geometric mean	6.690	6.621	7.238
Average ranks	1.967	1.833	2.200
Errors ratio	2.087	1.000	1.225
Significant wins	10/10	–	4/2
Wilcoxon test	0.3991	–	0.1332

Table A-3. Error rates of functional trees in classification problems. For the comparisons at the bottom, the reference algorithm is FT.

Dataset	Lin.Bayes	FT-U	FT-Leaves	FT-Inner	FT
Adult	-17.01 ± 0.49	14.18 ± 0.46	-14.31 ± 0.38	13.80 ± 0.39	13.83 ± 0.38
Australian	13.50 ± 0.27	-14.75 ± 0.96	-14.34 ± 0.43	13.94 ± 0.65	13.64 ± 0.58
Balance	-13.35 ± 0.35	-22.47 ± 1.1	-10.41 ± 0.63	7.555 ± 0.81	7.555 ± 0.81
Banding	23.68 ± 0.97	23.51 ± 1.8	23.64 ± 1.7	23.93 ± 2	23.93 ± 2
Breast(W)	$+2.862 \pm 0.093$	-5.123 ± 0.19	-4.337 ± 0.071	3.346 ± 0.39	3.346 ± 0.39
Cleveland	16.13 ± 0.39	-20.99 ± 1.4	$+15.95 \pm 0.49$	17.37 ± 0.9	16.67 ± 0.78
Credit	$+14.23 \pm 0.15$	14.61 ± 0.55	14.81 ± 0.5	15.12 ± 0.47	15.22 ± 0.55
Diabetes	$+22.71 \pm 0.21$	-25.35 ± 1	23.97 ± 1.0	-25.15 ± 0.93	23.57 ± 0.82
German	24.52 ± 0.23	-28.24 ± 0.73	$+23.63 \pm 0.51$	24.89 ± 0.55	24.4 ± 0.72
Glass	36.65 ± 0.8	$+32.15 \pm 2.3$	$+32.34 \pm 2.1$	35.18 ± 2.4	35.18 ± 2.4
Heart	17.70 ± 0.23	-23.07 ± 1.7	17.04 ± 0.58	17.33 ± 1.4	17.19 ± 0.8
Hepatitis	$+15.48 \pm 0.71$	17.13 ± 1.3	$+15.56 \pm 1.4$	17.17 ± 1.6	16.67 ± 1.3
Ionosphere	-13.38 ± 0.76	10.03 ± 0.93	10.79 ± 0.9	11.04 ± 1.6	11.04 ± 1.6
Iris	2 ± 0	-4.333 ± 0.79	2.067 ± 0.21	-3.867 ± 1.1	2.067 ± 0.21
Letter	-29.82 ± 1.3	11.88 ± 0.56	12.01 ± 0.58	11.83 ± 1.1	11.83 ± 1.1
Monks-1	-25.01 ± 0.030	10.54 ± 1.7	11.08 ± 1.6	9.045 ± 2	9.023 ± 2.0
Monks-2	-34.19 ± 0.58	-32.86 ± 0	-33.91 ± 0.41	9.143 ± 1.8	9.213 ± 1.8
Monks-3	-4.163 ± 0.011	$+1.572 \pm 0.39$	3.511 ± 0.87	2.792 ± 0.52	2.929 ± 0.52
Mushroom	-3.109 ± 0.037	$+0 \pm 0$	$+0.0616 \pm 0.006$	0.111 ± 0.039	0.111 ± 0.039
Optdigits	-4.687 ± 0.059	-9.476 ± 0.27	-4.73 ± 0.062	3.300 ± 0.13	3.304 ± 0.13
Pendigits	-12.43 ± 0.036	-3.559 ± 0.14	-3.097 ± 0.085	2.892 ± 0.086	2.892 ± 0.086
Pyrimidines	-9.846 ± 0.11	$+5.733 \pm 0.24$	6.105 ± 0.22	6.156 ± 0.17	6.158 ± 0.17
Satimage	-16.01 ± 0.13	-12.89 ± 0.23	-12.92 ± 0.22	11.78 ± 0.32	11.78 ± 0.32
Segment	-8.407 ± 0.09	3.381 ± 0.21	3.390 ± 0.21	3.208 ± 0.21	3.208 ± 0.21
Shuttle	-5.63 ± 0.3	0.0276 ± 0.025	0.0276 ± 0.025	0.036 ± 0.025	0.036 ± 0.025
Sonar	24.95 ± 1.2	27.65 ± 3.5	24.11 ± 1.7	24.69 ± 2.9	24.83 ± 2.7
Vehicle	22.16 ± 0.15	-27.33 ± 1.2	$+18.28 \pm 0.55$	21.03 ± 1.2	20.97 ± 1.2
Votes	-9.74 ± 0.21	3.773 ± 0.47	3.681 ± 0.41	3.703 ± 0.44	3.703 ± 0.44
Waveform	$+14.94 \pm 0.22$	-24.04 ± 0.8	$+15.22 \pm 0.20$	-16.18 ± 0.3	15.91 ± 0.44
Wine	1.133 ± 0.45	-6.61 ± 1.3	1.404 ± 0.29	1.459 ± 0.29	1.404 ± 0.29
Average mean	15.31	14.58	12.56	11.90	11.72
Geometric mean	11.632	9.741	6.978	6.800	6.621
Average ranks	3.500	3.483	2.617	2.833	2.567
Errors ratio	7.553	1.407	1.112	1.035	1.000
Significant wins	5/15	4/15	7/8	0/3	–
Wilcoxon test	0.00154	0.00154	0.724	0.055	–

Table A-4. Mean-squared error of linear regression and univariate trees in regression problems. For the comparisons at the bottom, the reference algorithm is FT-Univariate.

Dataset	FT Univ.	Lin.Regression	RT Univ.
Abalone	5.69 ± 0.093	$+4.908 \pm 0.0057$	$+5.402 \pm 0.069$
Auto	19.43 ± 1.2	$+11.47 \pm 0.12$	$+13.97 \pm 0.92$
Cart	0.9955 ± 0.00018	-5.684 ± 0.00068	-1.001 ± 0.0019
Computer	10.95 ± 0.65	-99.9 ± 0.25	11.45 ± 0.57
Cpu	4111 ± 1657	3734 ± 1717	2865 ± 1041
Diabetes	0.5355 ± 0.032	$+0.3994 \pm 0.028$	0.5531 ± 0.027
Elevators	$1.41e{-5} \pm 3.2e{-7}$	$+1.02e{-5} \pm 4.2e{-7}$	$1.46e{-5} \pm 5.2e{-7}$
Friedman	3.474 ± 0.018	-6.924 ± 0.00052	$+3.203 \pm 0.016$
House(16H)	$1.692e{+9} \pm 3.4e{+7}$	$-2.055e{+9} \pm 606768$	$+1.479e{+9} \pm 2.2e{+7}$
House(8L)	$1.193e{+9} \pm 1.2e{+7}$	$-1.731e{+9} \pm 81894$	$+1.134e{+9} \pm 1.3e{+7}$
House(Cal)	$3.691e{+9} \pm 3.5e{+7}$	$-4.809e{+9} \pm 2e{+6}$	$+3.516e{+9} \pm 3.8e{+7}$
Housing	19.59 ± 1.7	-23.84 ± 0.22	20.50 ± 2.1
Kinematics	0.03537 ± 0.0004	$-0.0408 \pm 1.5e{-5}$	-0.038 ± 0.0005
Machine	6036 ± 1752	5952 ± 2053	6535 ± 2556
Pole	48.55 ± 1.2	-930 ± 0.29	-54.96 ± 1.1
Puma32	$0.00011 \pm 1.5e{-6}$	$-0.00072 \pm 6.3e{-7}$	$+7.5e{-5} \pm 8.2e{-7}$
Puma8	13.31 ± 0.18	-19.93 ± 0.0077	$+11.20 \pm 0.087$
Pyrimidines	0.0142 ± 0.0015	-0.01811 ± 0.0031	0.01331 ± 0.0015
Hearth.QU	$0.03591 \pm 2.8e{-5}$	$+0.0358 \pm 3.9e{-5}$	$+0.03574 \pm 0.00018$
Triazines	0.01945 ± 0.0014	-0.02515 ± 0.0012	0.02044 ± 0.0013
Geometric mean	23.59	39.17	22.22
Average ranks	1.9	2.4	1.7
Errors ratio	1.0	2.94	0.95
Significant wins	–	13/5	3/9

Table A-5. Mean-squared error of model trees in regression problems. For the comparisons at the bottom, the reference algorithm is FT.

Dataset	FT	M5′	M5′NSmooth	RT.PLS
Abalone	4.608 ± 0.014	4.553 ± 0.5	4.67 ± 0.57	-4.963 ± 0.061
Auto	9.131 ± 0.52	7.958 ± 3.5	8.363 ± 4.2	-12.17 ± 0.5
Cart	1.012 ± 0.0012	0.9943 ± 0.017	0.9943 ± 0.017	-1.101 ± 0.0021
Computer	6.284 ± 0.58	8.081 ± 2.7	8.624 ± 4	-32.78 ± 0.94
Cpu	1070 ± 137	1092 ± 1315	1456 ± 1558	-2696 ± 642
Diabetes	0.3993 ± 0.028	0.4457 ± 0.32	0.4498 ± 0.32	0.3911 ± 0.036
Elevators	$5e{-}6 \pm 0$	$5.2e{-}6 \pm 1.4e{-}6$	$5e{-}6 \pm 6.7e{-}7$	$-1.07e{-}5 \pm 4.8e{-}7$
Friedman	1.85 ± 0.013	-1.938 ± 0.09	-1.954 ± 0.09	-2.749 ± 0.02
House(16H)	$1.231e9 \pm 2.2e7$	$1.27e9 \pm 1.1e8$	$-1.437e9 \pm 2.1e8$	$-1.308e9 \pm 1.5e7$
House(8L)	$1.015e9 \pm 1.3e7$	$997178776 \pm 7e7$	$1.095e9 \pm 1.2e8$	$-1.039e9 \pm 1.4e7$
House(Cal)	$3.053e9 \pm 3.1e7$	$3.070e9 \pm 2.8e8$	$-3.625e9 \pm 1.1e9$	$-3.199e9 \pm 2.3e7$
Housing	16.25 ± 1.3	12.47 ± 7.5	15.67 ± 9.7	16.07 ± 2.0
Kinematics	0.02318 ± 0.00027	-0.02548 ± 0.0015	-0.0276 ± 0.0018	0.02335 ± 0.00064
Machine	3032 ± 759	3557 ± 4271	4466 ± 5392	2492 ± 1249
Pole	79.3 ± 2.4	$+42 \pm 5.8$	$+52.81 \pm 11$	$+40.56 \pm 0.97$
Puma32	$8.24e{-}5 \pm 1.1e{-}6$	$+6.56e{-}5 \pm 3.2e{-}6$	$+6.83e{-}5 \pm 3.6e{-}6$	$8.3e{-}5 \pm 1.2e{-}6$
Puma8	11.24 ± 0.072	$+10.30 \pm 0.53$	$+10.37 \pm 0.58$	-11.92 ± 0.13
Pyrimidines	0.01296 ± 0.0031	0.01212 ± 0.024	0.01732 ± 0.036	0.01147 ± 0.0012
Heart.QU	0.03593 ± 0.00018	0.03571 ± 0.0052	0.03565 ± 0.0053	-0.03781 ± 0.00032
Triazines	0.02263 ± 0.0014	0.01746 ± 0.0071	0.01788 ± 0.0079	0.02288 ± 0.0012
Geometric mean	16.90	16.09	17.76	20.04
Average ranks	2.2	1.8	2.9	3.1
Errors ratio	1	0.98	1.07	1.37
Significant wins	–	2/3	4/3	12/1

Table A-6. Mean-squared error of functional trees in regression problems. For the comparisons at the bottom, the reference algorithm is FT.

Dataset	FT	FT.Leaves	FT.Inner	FT.Univ	Lin.Regre.
Abalone	4.608 ± 0.014	-4.753 ± 0.007	4.619 ± 0.029	-5.69 ± 0.093	-4.908 ± 0.0057
Auto	9.131 ± 0.52	9.587 ± 0.79	8.933 ± 0.38	-19.43 ± 1.2	-11.47 ± 0.12
Cart	1.012 ± 0.0012	$+0.9947 \pm 0.00015$	-1.016 ± 0.0017	$+0.9955 \pm 0.00018$	-5.684 ± 0.00068
Computer	6.284 ± 0.58	-6.507 ± 0.53	-6.426 ± 0.58	-10.95 ± 0.65	-99.9 ± 0.25
Cpu	1070 ± 137	-1197 ± 161	-1760 ± 389	-4111 ± 1657	-3734 ± 1717
Diabetes	0.3993 ± 0.028	0.4001 ± 0.027	-0.4998 ± 0.037	-0.5355 ± 0.032	0.3994 ± 0.028
Elevators	$5e{-}6 \pm 0$	$5e{-}6 \pm 0$	$-8.6e{-}6 \pm 5.2e{-}7$	$-1.41e{-}5 \pm 3.2e{-}7$	$-1.02e{-}5 \pm 4.2e{-}7$
Friedman	1.85 ± 0.013	-2.348 ± 0.015	1.85 ± 0.013	-3.474 ± 0.018	-6.924 ± 0.00052
House(16H)	$1.23e9 \pm 2.2e7$	$1.19e9 \pm 3e7$	$+1.2e9 \pm 2.1e7$	$-1.69e9 \pm 3.4e7$	$-2.06e9 \pm 606768$
House(8L)	$1.02e9 \pm 1.3e7$	$1.02e9 \pm 9156339$	$+1.01e9 \pm 1.3e7$	$-1.19e9 \pm 1.2e7$	$-1.73e9 \pm 818904$
House(Cal)	$3.05e9 \pm 3.1e7$	$+2.78e9 \pm 2.8e7$	$-3.09e9 \pm 2.7e7$	$-3.69e9 \pm 3.5e7$	$-4.81e9 \pm 2e6$
Housing	16.25 ± 1.3	$+13.36 \pm 1.7$	16.54 ± 1.3	-19.59 ± 1.7	-23.84 ± 0.22
Kinematics	0.023 ± 0.0003	-0.0258 ± 0.0004	-0.027 ± 0.0004	-0.035 ± 0.0004	$-0.041 \pm 1.5e{-}5$
Machine	3032 ± 759	-3300 ± 757	3473 ± 673	-6036 ± 1752	-5952 ± 2053
Pole	79.3 ± 2.4	$+35.16 \pm 0.72$	79.48 ± 2.6	$+48.55 \pm 1.2$	-930 ± 0.29
Puma32	$8.24e{-}5 \pm 1.1e{-}6$	$8.24e{-}5 \pm 1.1e{-}6$	$+7.15e{-}5 \pm 5.3e{-}7$	$-0.0001 \pm 1.5e{-}6$	$-0.0007 \pm 6.3e{-}7$
Puma8	11.24 ± 0.072	11.15 ± 0.084	$+11.05 \pm 0.087$	-13.31 ± 0.18	-19.93 ± 0.0077
Pyrimidines	0.0129 ± 0.003	0.0129 ± 0.003	$+0.0102 \pm 0.001$	0.0142 ± 0.002	-0.018 ± 0.003
Hearth.QU	0.0359 ± 0.0002	0.0359 ± 0.0002	$0.0359 \pm 6.7e{-}5$	$0.0359 \pm 2.8e{-}5$	$0.0358 \pm 3.9e{-}5$
Triazines	0.0226 ± 0.001	0.0227 ± 0.001	$+0.0179 \pm 8e{-}4$	$+0.0195 \pm 0.001$	-0.0252 ± 0.001
Geometric mean	16.90	16.48	17.69	23.59	39.17
Average ranks	2.02	2.28	2.4	3.9	4.4
Errors ratio	1	0.99	1.07	1.53	3.49
Significant wins	–	6/4	7/6	15/3	18/0

Table A-7. Error rates of bagging functional trees in classification problems. For the comparisons at the bottom, the reference algorithm is FT without bagging.

		Bagging		
Dataset	FT	FT-Leaves	FT	FT-Univ.
Adult	13.83 ± 0.38	14.41 ± 0.6	13.82 ± 0.64	-14.25 ± 0.35
Australian	13.64 ± 0.58	13.77 ± 5.2	13.35 ± 5.6	14.94 ± 5.6
Balance	7.555 ± 0.81	-10.24 ± 2.3	6.235 ± 2.8	-20.61 ± 3.9
Banding	23.93 ± 2	20.19 ± 5.3	20.55 ± 6.3	19.34 ± 5.4
Breast(W)	3.346 ± 0.39	4.419 ± 4.3	3.561 ± 2.9	6.268 ± 5.4
Cleveland	16.67 ± 0.78	16.51 ± 6.3	16.51 ± 6.3	20.85 ± 7
Credit	15.22 ± 0.55	15.37 ± 4.7	15.65 ± 4.7	15.23 ± 3.9
Diabetes	23.57 ± 0.82	23.18 ± 3.4	22.79 ± 2.7	25.25 ± 2.6
German	24.4 ± 0.72	23.5 ± 3.3	24.1 ± 4	-27.9 ± 2.8
Glass	35.18 ± 2.4	26.92 ± 12	33.09 ± 13	26.92 ± 12
Heart	17.19 ± 0.8	17.04 ± 4.7	17.04 ± 4.7	21.85 ± 9.3
Hepatitis	16.67 ± 1.3	14.19 ± 6	17.40 ± 6	16.7 ± 6
Ionosphere	11.04 ± 1.6	10.50 ± 5.3	10.82 ± 3.7	8.781 ± 3.8
Iris	2.067 ± 0.21	2 ± 3.2	2 ± 3.2	5.333 ± 6.9
Letter	11.83 ± 1.1	11.56 ± 0.79	$+10.27 \pm 0.7$	$+9.105 \pm 0.95$
Monks-1	9.023 ± 2.0	-23.43 ± 5.2	3.084 ± 8.2	-23.68 ± 5.6
Monks-2	9.213 ± 1.8	-39.14 ± 11	18.71 ± 18	-32.86 ± 0.65
Monks-3	2.929 ± 0.52	4.404 ± 5.8	3.933 ± 4.2	2.294 ± 3.4
Mushroom	0.1108 ± 0.039	0.0616 ± 0.065	0.1477 ± 0.2	$+0 \pm 0$
Optdigits	3.304 ± 0.13	-5.217 ± 1.2	3.419 ± 0.77	-7.407 ± 1.7
Pendigits	2.892 ± 0.086	2.574 ± 0.48	$+2.211 \pm 0.49$	2.846 ± 0.7
Pyrimidines	6.158 ± 0.17	$+5.475 \pm 0.7$	5.275 ± 1	$+5.046 \pm 1.1$
Satimage	11.78 ± 0.32	12.73 ± 2.9	11.75 ± 3	12.67 ± 2.7
Segment	3.208 ± 0.21	2.597 ± 1.0	2.727 ± 0.87	2.641 ± 1.0
Shuttle	0.0362 ± 0.025	0.0345 ± 0.024	0.0362 ± 0.03	0.0345 ± 0.024
Sonar	24.83 ± 2.7	23.99 ± 7.2	22.00 ± 7.9	19.68 ± 9.8
Vehicle	20.97 ± 1.2	19.95 ± 3.5	$+18.69 \pm 2.5$	-26.34 ± 3.4
Votes	3.703 ± 0.44	3.728 ± 4.9	3.496 ± 5	3.966 ± 4.9
Waveform	15.91 ± 0.44	14.92 ± 2.6	14.96 ± 2.3	-19.56 ± 2.6
Wine	1.404 ± 0.29	0.5556 ± 1.8	0.5556 ± 1.8	5.526 ± 5.2
Arithmetic mean	11.72	12.75	11.27	13.93
Geometric mean	6.62	6.86	6.15	9.52
Average rank	2.73	2.37	2.07	2.83
Significant wins	–	4/1	0/3	8/3
Wilcoxon test	–	0.51	0.0089	0.0641

Table A-8. Mean-squared error of bagging functional trees in regression problems. For the comparisons at the bottom, the reference algorithm is FT without bagging.

		Bagging		
Dataset	FT	FT-Leaves	FT	FT-Univ.
Abalone	4.608 ± 0.014	4.878 ± 3.3	4.704 ± 2.8	5.265 ± 3.3
Auto	9.131 ± 0.52	10.09 ± 7.7	8.832 ± 5.7	17.40 ± 13
Cart	1.012 ± 0.0012	0.9947 ± 0.021	1.008 ± 0.021	0.9957 ± 0.021
Computer	6.284 ± 0.58	6.455 ± 1.2	5.881 ± 1.3	6.98 ± 1.4
Cpu	1070 ± 137	2965 ± 5837	2601 ± 5134	4747 ± 8634
Diabetes	0.3993 ± 0.028	0.3688 ± 0.17	0.3790 ± 0.17	0.3797 ± 0.2
Elevators	$5e{-}6 \pm 0$	$6.163e{-}6 \pm 4.4e{-}6$	$6.161e{-}6 \pm 4.4e{-}6$	$-1.136e{-}5 \pm 6.3e{-}6$
Fried	1.85 ± 0.013	-1.972 ± 0.058	$+1.511 \pm 0.048$	-2.014 ± 0.054
House(16H)	$1.231e9 \pm 2.2e7$	$+996707700 \pm 5.5e7$	$+984746700 \pm 4.7e7$	$+1.086e9 \pm 7.9e7$
House(8L)	$1.015e9 \pm 1.3e7$	$+8.9e8 \pm 6.7e7$	$+8.75e8 \pm 7.2e7$	$+930206300 \pm 7.9e7$
House(Cal)	$3.053e9 \pm 3.1e7$	$4.417e9 \pm 2.2e9$	$3.623e9 \pm 1.3e9$	$-5.331e9 \pm 2.4e9$
Housing	16.25 ± 1.3	13.01 ± 4.4	13.35 ± 6.1	14.53 ± 5.6
H.Quake	0.03593 ± 0.00018	0.03628 ± 0.0051	0.03626 ± 0.0051	0.03743 ± 0.0059
Kinematics	0.02318 ± 0.00027	$+0.02113 \pm 0.0013$	$+0.01818 \pm 0.0010$	0.02203 ± 0.0013
Machine	3032 ± 759	5661 ± 8956	5297 ± 8629	6880 ± 12684
Pole	79.3 ± 2.4	$+31.15 \pm 4$	$+62.21 \pm 7$	$+28.75 \pm 4.4$
Puma32	$8.24e{-}5 \pm 1.1e{-}6$	$+6.983e{-}5 \pm 3.2e{-}6$	$+6.983e{-}5 \pm 3.2e{-}6$	$+6.612e{-}5 \pm 2.8e{-}6$
Puma8	11.24 ± 0.072	$+10.08 \pm 0.34$	$+10.26 \pm 0.36$	$+10.49 \pm 0.37$
Pyrimidines	0.01296 ± 0.0031	0.00991 ± 0.016	0.00991 ± 0.016	0.01029 ± 0.022
Triazines	0.02263 ± 0.0014	0.01939 ± 0.0077	0.01930 ± 0.0077	0.01594 ± 0.011
Geometric mean	16.90	16.97	16.69	19.01
Average rank	2.8	2.325	1.775	3.1
Significant wins	–	1/6	0/7	3/5

Acknowledgments

Thanks to the detailed comments of the editor and the anonymous reviewers that much improved the text. Gratitude is expressed to the financial support given by the FEDER, the Plurianual support attributed to LIACC, project APRIL, and project Adaptive Learning Systems—ALES (POSI/SRI/39770/2001).

Notes

1. At different nodes the system considers different number of classes depending on the class distribution of the examples that fall at this node.

2. We could penalize the degrees of freedom of each new attribute. In the actual implementation this has not been done.
3. The implementation of the framework allows both options. For strict linear models, as those we use in this work, this aspect is not relevant because a linear combination of linear combinations is also linear.
4. The LinearBayes classifier is a standard naive Bayes when all the attributes are nominal, and behaves like a linear discriminant when all the attributes are continuous. For mixed attribute types LinearBayes makes a fusion of these classifiers. LinearBayes generates a naive Bayes model using the nominal attributes, and a discriminant function using the continuous ones. To classify a test example, LinearBayes sums the class probability distributions given by both, and selects the class that maximizes the sum.
5. We have used M5′ from version 3.1.8 of the Weka environment.
6. Each pair of data points consists of the estimate statistic on one dataset and for the two learning algorithms being compared.
7. http://www.niaad.liacc.up.pt/∼ltorgo/Datasets.
8. In regression problems, the actual implementation ignores missing values at learning time. At application time, if the value of the test attribute is unknown, all descendant branches produce a prediction. The final prediction is a weighted average of the predictions.

References

Berthold, M., & Hand, D. (1999). *Intelligent data analysis—An introduction*. Springer Verlag.

Bhattacharyya, G., & Johnson, R. (1977). *Statistical concepts and methods*. New York: John Willey & Sons.

Blake, C., Keogh, E., & Merz, C. (1999). UCI repository of machine learning databases.

Brain, D., & Webb, G. (2002). The need for low bias algorithms in classification learning from large data sets. In T. Elomaa, H. Mannila, & H. Toivonen (Eds.), *Principles of data mining and knowledge discovery PKDD-02*, LNAI 2431 (pp. 62–73). Springer Verlag.

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*, 123–140.

Breiman, L. (1998). Arcing classifiers. *The Annals of Statistics, 26:3*, 801–849.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees*. Wadsworth International Group.

Brodley, C. E. (1995). Recursive automatic bias selection for classifier construction. *Machine Learning, 20*, 63–94.

Brodley, C. E., & Utgoff, P. E. (1995). Multivariate decision trees. *Machine Learning, 19*, 45–77.

Frank, E., Wang, Y., Inglis, S., Holmes, G., & Witten, I. (1998). Using model trees for classification. *Machine Learning, 32*, 63–82.

Frank, E., & Witten, I. H. (1998). Generating accurate rule sets without global optimization. In J. Shavlik (Ed.), *Proceedings of the 15th international conference—ICML'98* (pp. 144–151). Morgan Kaufmann.

Gama, J. (1997). Probabilistic linear tree. In D. Fisher (Ed.), *Machine learning, Proc. of the 14th international conference* (pp. 134–142). Morgan Kaufmann.

Gama, J. (2000). A linear-bayes classifier. In C. Monard, & J. Sichman (Eds.), *Advances on artificial intelligence—SBIA2000*, LNAI 1952 (pp. 269–279). Springer Verlag.

Gama, J., & Brazdil, P. (2000). Cascade generalization. *Machine Learning 41*, 315–343.

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilema. *Neural Computation, 4*, 1–58.

Ihaka, R., & Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics, 5:3*, 299–314.

Karalic, A. (1992). Employing linear regression in regression tree leaves. In B. Neumann (Ed.), *European conference on artificial intelligence* (pp. 440–441). John Wiley & Sons.

Kim, H., & Loh, W. (2001). Classification trees with unbiased multiway splits. *Journal of the American Statistical Association, 96*, 589–604.

Kim, H., & Loh, W.-Y. (2003). Classification trees with bivariate linear discriminant node models. *Journal of Computational and Graphical Statistics, 12:3*, 512–530.

Kohavi, R. (1996). Scaling up the accuracy of naive-bayes classifiers: A decision tree hybrid. In *Proc. of the 2nd international conference on knowledge discovery and data mining* (pp. 202–207). AAAI Press.

Kohavi, R., & Wolpert, D. (1996). Bias plus variance decomposition for zero-one loss functions. In L. Saitta (Ed.), *Machine learning, Proc. of the 13th international conference* (pp. 275–283). Morgan Kaufmann.

Kononenko, I., Cestnik, B., & Bratko, I. (1988). *Assistant professional user's guide*. Technical report, Jozef Stefan Institute.

Li, K. C., Lue, H., & Chen, C. (2000). Interactive tree-structured regression via principal Hessians direction. *Journal of the American Statistical Association, 95*, 547–560.

Loh, W., & Shih, Y. (1997). Split selection methods for classification trees. *Statistica Sinica, 7*, 815–840.

Loh, W., & Vanichsetakul, N. (1988). Tree-structured classification via generalized discriminant analysis. *Journal of the American Statistical Association, 83*, 715–728.

McLachlan, G. (1992). *Discriminant analysis and statistical pattern recognition*. New York: Willey and Sons.

Mitchell, T. (1997). *Machine learning*. MacGraw-Hill Companies, Inc.

Murthy, S., Kasif, S., & Salzberg, S. (1994). A system for induction of oblique decision trees. *Journal of Artificial Inteligence Research, 2*, 1–32.

Perlich, C., Provost, F., & Simonoff, J. (2003). Tree induction vs. logistic regression: A learning-curve analysis. *Journal of Machine Learning Research, 4*, 211–255.

Quinlan, R. (1992). Learning with continuous classes. In Adams, & Sterling (Eds.), *5th Australian joint conference on artificial intelligence* (pp. 343–348). World Scientific.

Quinlan, R. (1993a). *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers, Inc.

Quinlan, R. (1993b). Combining instance-based and model-based learning. In P. Utgoff (Ed.), *Machine learning, proceedings of the 10th international conference* (pp. 236–243). Morgan Kaufmann.

Sahami, M. (1995). Generating neural networks through the induction of threshold logic unit trees. In *Proceedings of the first international IEEE symposium on intelligence in neural and biological systems* (pp. 108–115). IEEE Computer Society.

Seewald, A., Petrak, J., & Widmer, G. (2001). Hybrid decision tree learners with alternative leaf classifiers: An empirical study. In *Proceedings of the 14th FLAIRS conference* (pp. 407–411). AAAI Press.

Todorovski, L., & Dzeroski, S. (2003). Combining classifiers with meta decision trees. *Machine Learning, 50*, 223–249.

Torgo, L. (1997). Functional models for regression tree leaves. In D. Fisher (Ed.), *Machine learning, proceedings of the 14th international conference* (pp. 385–393). Morgan Kaufmann.

Torgo, L. (2000). Partial linear trees. In P. Langley (Ed.), *Machine learning, proceedings of the 17th international conference* (pp. 1007–1014). Morgan Kaufmann.

Utgoff, P. (1988). Percepton trees—A case study in hybrid concept representation. In *Proceedings of the seventh national conference on artificial intelligence* (pp. 601–606). AAAI Press.

Utgoff, P., & Brodley, C. (1991). *Linear machine decision trees*. Coins technical report, 91-10, University of Massachusetts.

Witten, I., & Frank, E. (2000). *Data mining: Practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann Publishers.

Wolpert, D. (1992). Stacked generalization. *Neural Networks* (vol. 5, pp. 241–260). Pergamon Press.

Received September 3, 2002
Revised January 19, 2004
Accepted January 20, 2004
Final manuscript January 20, 2004