Bias - variance tradeoff of soft decision trees

Cristina Olaru
University of Liège
Montefiore Institute B28
B4000, Liège, Belgium
olaru@montefiore.ulg.ac.be

Louis Wehenkel
University of Liège
Montefiore Institute B28
B4000, Liège, Belgium
L.Wehenkel@ulg.ac.be

Abstract

This paper focuses on the study of the error composition of a fuzzy decision tree induction method recently proposed by the authors, called soft decision trees. This error may be expressed as a sum of three types of error: residual error, bias and variance. The paper studies empirically the tradeoff between bias and variance in a soft decision tree method and compares it with the tradeoff of classical crisp regression and classification trees. The main conclusion is that the reduced prediction variance of fuzzy trees is the main reason for their improved performance with respect to crisp ones.

Keywords: Fuzzy Decision Tree, Classification, Bias, Variance

1 Introduction

Fuzzy decision tree induction is a combination between fuzzy reasoning and automatic learning based on decision tree induction. It is a model concerned with the (automatic) design of *fuzzy if-then* rules in a tree structure. It is used in classification problems ([1, 7, 8, 13]) in most of the cases but sometimes also in regression problems ([11]).

The introduction of the fuzzy environment in the decision tree induction technique has been suggested within the fuzzy decision tree community for three reasons: i) in order to enlarge the use of decision trees towards the ability to *manage fuzzy information* in the form of fuzzy inputs, fuzzy classes, and fuzzy rules manipulation, ii) in order to *improve their predictive accuracy* and respectively iii) in order to obtain *reduced model complexity.*

Building a fuzzy decision tree model is in most of the literature solely a question of tree *growing.* We speak about a top-down induction either as a crisp tree that afterwards is fuzzified, or as a tree that directly integrates fuzzy reasoning during the growing phase. The numerical inputs must be fuzzified (transformed from continuous values to fuzzy values) and this fuzzification step is rarely automatic. There are approaches that consider also a *pruning* stage after growing in order to tradeoff better the model to data. A few authors have implemented after growing, also a global optimization phase of some parameters of the model. A global optimization aims at tuning the parameters of the model not anymore based on local learning samples but on the whole learning set and the already identified structure of the tree.

We have proposed, explained and validated in reference [9] a complete fuzzy decision tree technique, called soft decision tree induction. The main objective of the present paper is to verify quantitatively the conjecture that the accuracy improvement of soft versus crisp trees is essentially a consequence of the reduced variance of soft trees with respect to crisp ones.

The paper is organized as follows. Section 2 explains intuitively the soft decision tree (SDT) method. Section 3 defines the bias-variance tradeoff in a SDT. Section 4 presents the experimental results concerning the bias-variance tradeoff in a SDT and section 5 ends the paper with some conclusions.

2 Soft decision tree induction

A soft decision tree is a method able to partition the input space into a set of rectangles and then approximate the output in each rectangle by a smooth curve, instead of a constant or a class like in the case of crisp tree-based methods. A soft tree is an approximation structure to compute the degree of membership of objects to a particular class (or concept) or to compute a numerical output of objects, as a function of the attribute values of these objects. The goal is to recursively split the input space into (overlapping) sub-

regions of objects which have the same membership degree to the target class (in the case of classification problems) or the same output value (in the case of regression problems). As in any tree-based approach, the hypothesis space of a soft decision tree model is a family of structures. A soft decision tree structure is determined by the graph of the tree and by the attributes attached to its test nodes. The parameters in all the test nodes together with the labels of all the terminal nodes represent the parameters of the tree-based model. There is a search over both structure and parameter spaces so as to learn a model from experience.

2.1 Overview of the method

We now give an overview of our SDT method[1]. The process of building a soft decision tree starts by *growing* a "sufficiently large" tree using a set of objects called growing set GS. Tree nodes are successively added in a top-down fashion, until stopping criteria are met. Then the grown tree is *pruned* in a bottom-up fashion to remove its irrelevant parts. At this stage, the hold-out technique is used which makes use of an another set of objects, called the pruning set PS. Next, a third step could be either a *refitting* step or a *backfitting* step. Both consist of tuning certain parameters of the pruned tree model in order to improve its approximation capabilities further. These steps may use the whole learning set $LS = GS \cup PS$ or only the growing set GS. At the end of every intermediate stage, the obtained trees (fully developed, pruned, refitted or backfitted) may be *tested*. A third sample, independent from the learning set, called test set TS, is used to evaluate the predictive accuracy of these trees.

2.2 Soft tree semantics

Let us remind that a fuzzy set $S \subset U$ of objects is characterized by a *membership function* $\mu_S : U \rightarrow [0,1]$ which associates with each object o of the universe of objects U a number $\mu_S(o)$ in the interval $[0,1]$ representing the degree of affiliation of the object o to S. Let us denote by C the output class, be it crisp (0/1) or fuzzy, by $\mu_C(o)$ the degree of membership of an object o to this class, and by $\hat{\mu}_C(o)$ this membership degree as estimated by a tree. We assume that all the attribute values are numerical and normalized in [0,1].

In a soft decision tree each node of the tree is a fuzzy set. A discriminator function $\nu \rightarrow [0,1]$ is associated to each test node. This discriminator function determines the node's fuzzy dichotomy based on the chosen attribute a at the node and several parameters. We restrict in this paper to piecewise linear discriminator templates (see figure 1). These are defined by a threshold α and width parameter β. All the objects of the fuzzy set node S corresponding to the condition $a(o) \leq \alpha + \frac{\beta}{2}$ are directed towards the left successor S_L with the membership degree given by

$$\mu_{S_L}(o) = \mu_S(o)\nu(a(o), \alpha, \beta). \qquad (1)$$

Similarly, the objects directed towards the right successor S_R are weighted by the complement of the discriminator function value

$$\mu_{S_R}(o) = \mu_S(o)(1 - \nu(a(o), \alpha, \beta)) \qquad (2)$$

and correspond to the condition $a(o) \geq \alpha - \frac{\beta}{2}$.

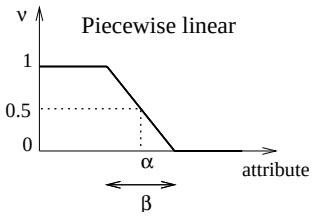

Figure 1: The piecewise linear discriminator function

Let j denote a node of a tree, let L_j denote a numerical value (or label) attached to this node, and let S_{L_j} be the fuzzy subset corresponding to this node. Then, the membership degree to C estimated by the soft tree for a certain object o is the average of all the labels L_j attached to the leaves, weighted by the membership degrees of the object to the fuzzy subsets of these leaves, $\mu_{S_{L_j}}(o)$:

$$\hat{\mu}_C(o) = \frac{\sum_{j \in leaves} \mu_{S_{L_j}}(o) L_j}{\sum_{j \in leaves} \mu_{S_{L_j}(o)}}, \qquad (3)$$

where *leaves* denotes the set of indexes of the terminal nodes of the tree. Notice that $\sum_{j \in leaves} \mu_{S_{L_j}(o)}$ is equal to the degree of membership of the object to the root node $\mu_R(o)$, and is normally equal to 1.

2.3 Building a soft decision tree

The *growing* procedure of a SDT follows the same principles as the crisp decision tree induction, only the partitioning procedure is different, being a fuzzy partitioning and not anymore a crisp partitioning. In our method, its objective is to determine in a node S the best attribute a for splitting, the cutting point parameter α, the degree of fuzziness β of

[1] To save space we recall only the basics and kindly ask the reader to refer to reference [9] for details.

the fuzzy partitioning and the labels of the two successors of the node, L_L and L_R. For this, an error function of squared error type adapted to the fuzzy framework is minimized

$$E_S = \sum_{o \in S} \mu_S(o)[\mu_C(o) - \hat{\mu'}_C(o)]^2 \qquad (4)$$

where

$$\hat{\mu'}_C(o) = \nu(a(o), \alpha, \beta)L_L + (1 - \nu(a(o), \alpha, \beta))L_R. \qquad (5)$$

The strategy consists in decomposing the search into two parts: first, searching for the attribute a and threshold α at fixed null β, as if the node would be crisp; for this, strategies adapted to the fuzzy case from CART [2] regression trees are used; second, with the found attribute and threshold α kept frozen, searching for width β by Fibonacci search, and for every β value, updating labels in the successor nodes, L_L and L_R, by linear regression formulas.

Next, the *pruning* stage aims at finding a subtree of the large tree obtained at the growing stage, which presents the best mean absolute error (MAE) computed on the pruning set: $\arg\min_{\text{subtrees of SDT}} MAE_{PS}$.

The strategy consists of three steps: firstly, all the test nodes of the full tree are sorted by increasing order of their relevance to the tree error, secondly, the nodes from the previous list are removed one by one in the established order and the result is a sequence of subtrees. And finally, the best subtree in terms of error is selected by the so-called "one-standard-error-rule".

Then, the objective of the *refitting* step, given a tree and a refitting set RS of objects is to optimize the labels $q_j = L_j$ in all the terminal nodes by minimizing the squared error function

$$E(q) = ||Y - \hat{Y}||^2, \qquad (6)$$

where $\hat{Y} = Mq$ in conformity with eq. (3), $Y_i = \mu_C(o_i)$, $\hat{Y}_i = \hat{\mu}_C(o_i)$ and $M_{ij} = \mu_{S_{L_j}}(o_i)$ with $i = 1 \ldots ||RS||$, $j = 1 \ldots K+1$, K being the number of tree test nodes. Refitting is a linear regression optimization problem, in practice fast and efficient in terms of accuracy.

A more complex optimization stage is *backfitting* whose objective, given a tree and a backfitting set, is to find not only the labels in terminal nodes but also the parameters α and β in all the test nodes, by minimizing the squared error function

$$E(q) = \sum_{o \in BS} (\mu_C(o) - \hat{\mu}_C(o, q))^2, \qquad (7)$$

where q denotes all the free parameters of the SDT. The strategy is to use Levenberg-Marquardt non-linear optimization technique, where only partial derivatives have to be computed, not also second order derivatives. The delicate part of the optimization stage is the computation of the partial derivatives. We developed for this purpose a kind of back-propagation algorithm linear in the tree complexity (see [9] for details). Nevertheless, the backfitting process is more time consuming than refitting.

Growing and pruning steps are a phase of structure identification using a local approach (which means node by node), whereas refitting and backfitting are optimizations of the model parameters done in a global manner (which means all the nodes at the same time).

3 Bias-variance tradeoff

If we consider a regression learning algorithm, the mean squared prediction error of an automatic learning method may be expressed as a sum of three types of error: residual error, bias and variance.

$$MSE_{algo} = resid\_err + bias^2_{algo} + var_{algo}. \qquad (8)$$

The residual error $resid\_err$ represents the expected error of the Bayes model, thus the minimum error one can get by training a model of unrestricted complexity, hypothesis space and amount of data concerning the learning problem. It reflects the irreducible prediction error and is beyond control. The bias term $bias^2_{algo}$ reflects the persistent or systematic error that the learning model is expected to have on average when trained on learning sets of the same finite size and is independent of the learning sample. Generally, the simpler the model, the higher the bias. The variance term var_{algo} reflects the variability of the model induced by the randomness of the learning set used. It reflects the sensitivity of the model estimate to the learning set. A too complex model is likely to have a high variance and the model may exhibit what we call an *overfitting* problem, an over adjustment of the model parameters to the learning data.

From eq. (8) one may remark the interest in reducing as much as possible both bias and variance terms, given that the term of the residual error cannot be reduced for a given problem. Since generally, in decision tree models, reducing variance increases bias

and vice versa, there is a *bias-variance tradeoff* in order to improve the error prediction of the models.

In the case of crisp decision and regression trees, many experimental studies [6] have shown that variance is the most important source of error in the vast majority of problems. Even a small change of the learning sample may result in a very different tree and this results in a high variance and hence small accuracy.

There are mainly two ways to reduce the variance in decision trees and obtain a better tradeoff. Firstly, there are different ways to reduce the model complexity or to control it. Pruning is such a way. Its serious drawback is that it does not succeed always to improve the accuracy, because together with the variance reduction appears also a bias increase. Secondly, recently, aggregation methods are well established as a way to obtain highly accurate classifiers by combining less accurate ones, but unfortunately loosing also in interpretability. The most well known aggregation techniques are bagging and boosting. We will show in this paper that a third possible way to reduce variance in decision trees is the soft decision tree induction method.

3.1 Bias and variance estimation of SDTs

We consider that a learning set LS used in order to construct a soft decision tree is a sample of independent and identically distributed objects drawn from the same probability space. Let us then denote by $\hat{y}_{LS}(o)$ the output estimated by a SDT built from a random learning set LS of size N at a point $o \in U$ of the universe U of objects. Then the *global variance* of the SDT learning algorithm can be written as

$$var_{SDT} = E_U\{E_{LS}\{(\hat{y}_{LS}(o) - E_{LS}\{\hat{y}_{LS}(o)\})^2\}\} \quad (9)$$

where the innermost expectations are taken over the distribution of all learning sets of size N and the outermost expectations over the distribution of objects.

The *global bias* of the SDT algorithm reflects the systematic error that the SDT is having in average with respect to the Bayes model when trained on learning sets of the same finite size. The Bayes model represents the best possible model one can get for a learning problem according to a mean squared error minimization. Since for the analysed problems (datasets), the Bayes model is not known a priori, the bias of the SDT algorithm cannot be calculated separately from the residual error only from data. However, in order to study the global bias *relative* evolution, we compute the sum of the residual error and bias terms, since the residual error is a constant term:

$$bias^2_{SDT} + resid\_err = E_U\{(y(o) - E_{LS}\{\hat{y}_{LS}(o)\})^2\}. \quad (10)$$

Denoting by α_{LS} the threshold parameter at a given node of a SDT built from a random learning set LS of size N, the *parameter variance* of the SDT reflects the variability of the parameter induced by the randomness of the learning set used. It can be written as

$$var_\alpha = E_{LS}\{(\alpha_{LS} - E_{LS}\{\alpha_{LS}\})^2\}. \quad (11)$$

Experiments. In order to compute the expectations over LS, E_{LS}, we should draw an infinite number of learning sets of the same size and build $SDTs$ on them. We make the compromise of randomly sampling without replacement the available learning set LS into a finite number of q learning samples LS_i of the same size, $i = 1, \ldots, q$. Notice that the size of the sub-samples LS_i should be significantly smaller than the size of the LS from which they are drawn. For the expectation over the input space E_U we choose to sum up over the whole test set TS.

4 Experimental study

4.1 Datasets

We show in this section results obtained on 3 datasets: gaussian, twonorm and omib. Gaussian [6] is a synthetic database with 20000 objects, 2 classes and 2 attributes. Each class corresponds to a bi-dimensional Gaussian distribution. The first Gaussian distribution is centered at (0.0;0.0) and has a diagonal covariance matrix, while the second one is centered at (2.0;2.0) and has a non-diagonal covariance matrix. There is an important overlapping between the two classes. The Bayes classifier is of quadratic shape.

Twonorm dataset [3] is a database with 2 classes and 20 attributes. There are 10000 objects generated. Each class is drawn from a multivariate normal distribution with unit covariance matrix. One class has mean (a,a,...a) while the other class has mean (-a,-a,...-a) where $a = \frac{2}{\sqrt{20}}$. The optimal separating surface (Bayes classifier) is an oblique plane, hard to approximate by the multidimensional rectangles used in a crisp tree.

The omib[2] database [12] is an electric power system database with 20000 objects that does a transient security assessment task. A three-phase short-circuit occurs in the system, normally cleared after 155ms. The problem output reflects the system security state after

[2]The acronym OMIB stands for One-Machine-Infinite-Bus system.

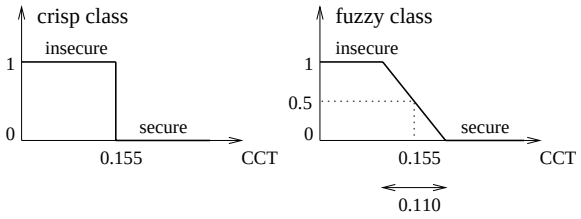

Figure 2: The crisp class and the fuzzy class for OMIB data

the short-circuit fault occurrence. The input space is defined by 6 attributes representing pre-fault operating conditions of the OMIB system. The continuous output is defined as the fuzzy or crisp class of figure 2. The problem is deterministic, that is why the Bayes error rate (the residual error) is null.

4.2 Protocol of experiments

Each database was split into a pool set, a pruning set PS and a test set TS. From the pool set, multiple q growing sets GS of the same size have been randomly chosen by sampling without replacement in order to build multiple models for each dataset. Table 1 presents the size of each set: pool set, growing set, pruning set and test set for the three datasets. Each model has been grown on GS, pruned on PS, tested on TS, and refitted or backfitted on GS. In the results presented further $q = 25$ models have been averaged for the study of the global variance and $q = 20$ for the study of the parameter variance. The soft decision trees have been compared with CART regression trees [2] and ULG decision trees [12]. CART and ULG trees were completely grown and then pruned. CART and SDT were trained for a regression goal (0/1 in the case of the crisp pre-classified datasets, the membership degree to the target class μ_C in the case of fuzzily classified ones). ULG was trained on a crisp classification goal, classes being converted into 0/1 class-membership degrees to compute bias, variance and mean squared errors.

Table 1: Datasets

Database	Pool set	GS	PS	TS
Gaussian	14000	100	2000	4000
Twonorm	7000	300	1000	2000
Omib	14000	500	2000	4000

4.3 Comparing SDT bias-variance tradeoff with crisp methods tradeoff

The global variance and bias [3] have been computed for each method for the GS size of table 1. Figure 3 shows for each dataset the proportion of bias and variance in the mean squared error (MSE) of each method: CART, ULG, fully grown (F), pruned (P), refitted (R) and backfitted (B) SDTs. We may notice the next aspects from these graphics: SDT in all its versions, improves accuracy with respect to a crisp tree, being regression or decision tree, SDTs reduce very much the variance of crisp trees, pruning step succeeds to leave slightly constant bias and variance quantities and refitting and backfitting steps increase variance of SDTs.

[3] What we will call further bias is in fact the relative bias of eq. (10).

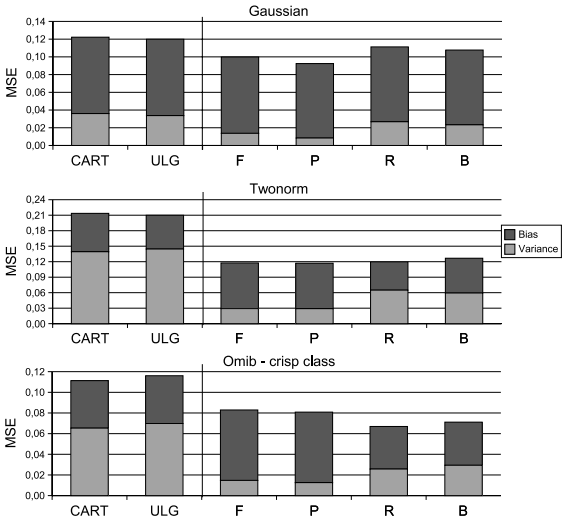

Figure 3: The proportion of variance and bias in the mean squared error (MSE)

4.4 Variance and bias variation in terms of the model complexity

We computed bias, variance and error quantities for models of different complexities. The complexity of a tree model is given here by the number of its test nodes. Figure 4 displays the evolution of MSE, variance and bias with the model complexity, for twonorm database, for fully-grown CART and fully-grown SDT. For a crisp tree as CART, the evolution is the classic one: bias decreases and variance increases with the complexity. And there is a trade-off point where the mean squared error is the best and the corresponding complexity is the complexity of the right sized tree: about 8 test nodes. We expected to find the same type of curve evolutions for SDTs but the result looks like this: the variance is almost constant and very low, the bias is the principal source of error, and the error always decreases with the model complexity. Thus, the more complex a SDT, the better the accuracy. As the figures show, we point out that we found the tradeoff point between bias and variance by scanning CART regression trees up to 25 test nodes complexity. Whereas for the SDT, models with up to 350 test nodes have been

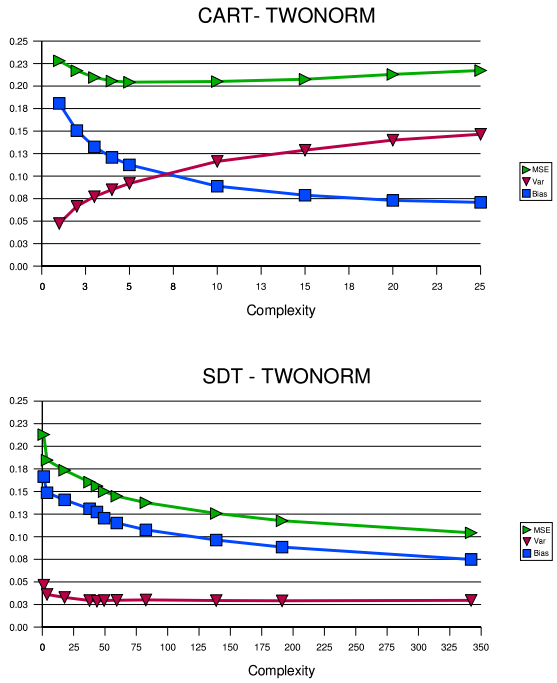

Figure 4: Evolution of MSE, variance and bias with the model complexity

scanned and no such point has been found.

4.5 Reason for the reduced variance

The explanation of why a soft decision tree presents a low variance regardless of the model complexity just after the growing step lies in the possibility of overlapping instances during each tree node partitioning. One aspect of the recursive partitioning of crisp decision trees is the fast decrease of local growing samples when going down into the tree. The local decisions may become too particular to the data, thus leading to overfitting and to high variance [4]. In soft decision trees, the local growing sets are allowed to keep more instances, even if the instances are not all strictly belonging to those sets. The larger the degree of fuzziness in a node, the larger the local growing sets of its successors. Thus, local decisions are more stable in a node as they are established on richer information, all the parameters in nodes are chosen based on more information and the variance linked to this aspect is less significant.

4.6 CPU time complexity

The drawback of the improved accuracy of soft decision trees is their increase in CPU time needs. Table 2 shows an example of a single typical run CPU time for the tested algorithms, obtained on a 1.0GHz, 512MB Linux Pentium III. The methods are programmed all in Common Lisp programming language and the CPU time exclude garbage-collecting. We notice a big gap between SDTs and the crisp trees. This is the price paid for obtaining a more accurate but still interpretable tool.

Table 2: CPU time (in seconds)

Dataset	ULG	CART	F	P	R	B
Gaussian	5	3	32	34	38	50
Twonorm	5	2	54	61	67	406
Omib	6	6	41	50	65	122

4.7 Fuzzy versus crisp output definition

One very interesting property of our soft decision tree algorithm is that it can reproduce a classical crisp classification but also a fuzzy output when such a fuzzy membership degree to the class is given in the dataset. To show the potential of our method in this case, we have chosen a database where it was possible to define a fuzzy output because we had the necessary knowledge about the problem: the OMIB stability problem. Figure 2 shows the definitions of both crisp and fuzzy classes. They are both defined based on a parameter called CCT, the critical clearing time of the disturbance, which may be interpreted as a degree of stability. So if the objective is to classify situations with respect to the threshold of 155ms, we actually have two possibilities: either we build a (crisp or soft) tree directly with respect to the discrete 0/1 output, or we use the fuzzy output to build a tree and then discretize its output a posteriori. Figure 5 represents the mean squared error for CART, fully grown (F), pruned (P), refitted (R) and backfitted (B) SDTs, in the crisp output case definition and in the fuzzy output case. As we may notice, all the quantities, MSE, bias and variance, are much decreased by the use of fuzzy output, for all the algorithms. So, when a fuzzy output is available, it can thus be used to improve the accuracy of the algorithms.

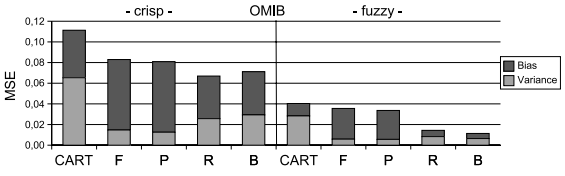

Figure 5: Crisp (left part) versus fuzzy (right part) class on CART and SDT results

4.8 Variance and bias as function of the growing set size

Figure 6 displays the evolution of MSE, variance and bias with the growing set size on

omib fuzzy output dataset for the backfitted version of SDTs. As expected, both bias and variance decrease with the growing set size. They are almost constant and equal quantities for medium and large sample sizes. The refitted version of the SDTs gives the same allure of the curves as the backfitted SDT for all the datasets investigated.

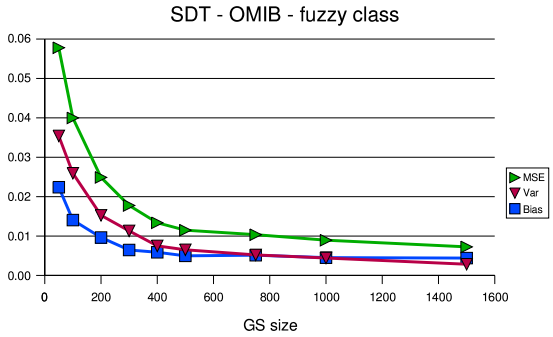

Figure 6: Evolution of MSE, variance and bias with the growing set size

4.9 Parameter variance

Some references [5] show that classical discretization methods of decision trees actually lead to very high variance of the cutting point parameter, even for large learning sample sizes. We studied the behaviour of SDTs from this point of view by measuring the variance of the cutting point parameter in a test node of the tree: firstly, in the root node, and secondly, in the second level test nodes (at depth 2). The procedure was like follows. For each growing set size, we built q trees (CART, ULG and SDT). For each of these trees we kept the α cutting point parameter of the root node of the tree given that was corresponding to the same attribute. Thus we obtained at most q parameters, since not necessarily all the trees have chosen the same attribute in the root node. We computed the variance on these parameters.

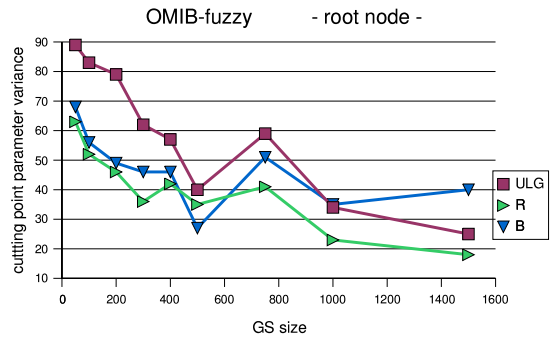

Figure 7: Evolution of the parameter variance with the growing set size at the root node

Figure 7 displays the evolution of the cutting point parameter variance with the growing set size at the root node, for the fuzzy class definition of the omib database, for ULG decision trees, refitted (R) and backfitted (B) SDTs. For this dataset, all the trees have chosen naturally the same attribute in the root node. Due to the adopted fuzzy partitioning approach, the chosen attribute in the root node of a non-backfitted soft decision tree and its α value will always coincide with the ones chosen in the root node of a CART regression tree. For this reason, the cutting point variance is identical for CART and refitted SDT in the root node. Once backfitted, a soft decision tree changes its thresholds in all the tree nodes, and thus also its parameter variance. One may observe from figure 7 that a non-backfitted soft decision tree, identically to a CART regression tree, presents less parameter variance in the root node than a ULG decision tree. By backfitting, parameter variance in a root node of a SDT increases with respect to the non-backfitted version. The explanation resides in the fact that by globally optimizing, the location parameters are not any more restricted to fall in the range [0,1] and therefore they are more variable with respect to the average.

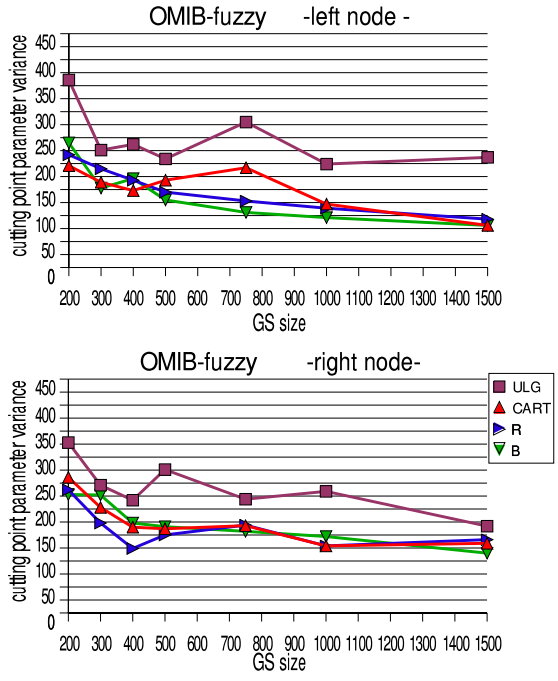

Figure 8: Evolution of the parameter variance with the growing set size at the second level nodes (depth 2)

By studying the parameter variance in the root node of a soft decision tree, we are not able to see the contribution of the fuzzy approach of splitting to the parameter variance in a whatever internal test node. Thus, we did

also an experiment where the structure of the SDT was fixed and the complexity was settled to 3 test nodes. The structure has been determined with an ULG decision tree built with the whole dataset. We present here the study for omib fuzzy output dataset. We may notice from figure 8 that CART, refitted SDT and backfitted SDT do not seem to show a significant difference in their parameter variance at the second level nodes, but they are definitely all more stable from this point of view than the ULG method. The conclusion of this study is that the reduction in the global variance of SDTs is not correlated with a reduction in the parameter variance (with respect to CART). Thus, the fuzzy partitioning does not improve the parameter stability with respect to a crisp partitioning.

5 Conclusions

This paper studied bias and variance of soft decision trees. An important conclusion of our experiments is that the improved accuracy of soft decision trees with respect to crisp decision and regression trees is explained essentially by a lower prediction variance. Thus, soft decision trees are indeed another tool for reducing variance, beside pruning and aggregation techniques. However, complementary studies [10] have shown that soft decision trees do not reduce variance as much as aggregation methods do. We have pointed out also that the variance in soft decision trees does not increase with the model complexity and tends to remain constant for large SDT models, as opposite to the case of crisp trees which variance increases quickly with their complexity, and we have argue on the cause of this behaviour. Some hints regarding the time complexity of each method have been provided. We have also highlighted the possibility to exploit a fuzzy pre-classification of objects and shown that this may also very significantly improve accuracy. Finally, the parameter variance study showed that the reduction in prediction variance of a soft decision tree with respect to a crisp regression tree is not correlated with a reduction in parameter variance.

References

[1] X. Boyen and L. Wehenkel. Automatic induction of fuzzy decision trees and its applications to power system security assessment. *Fuzzy Sets and Systems*, 102 (1999) 3-19.

[2] L. Breiman, J. H. Friedman, R. A. Olshen and C.J. Stone. Classification and regression trees. Chapman and Hall, New-York, 1984.

[3] L. Breiman. Arcing Classifiers. *The Annals of Statistics*. Volume 26, Number 3, pages 801-849, 1998.

[4] J.H. Friedman. Local learning Based On Recursive Covering. Technical report. Department of Statistics, Standford University, August 1996.

[5] P. Geurts and L. Wehenkel. Investigation and Reduction of Discretization Variance in Decision Tree Induction. In *ECML 2000, Proceedings of 11th European Conference on Machine Learning*, Barcelona, Spain, May/June, pages 162-170, 2000.

[6] P. Geurts. Contributions to Decision Tree Induction: Bias/Variance Tradeoff and Time Series Classification. PhD Thesis, University of Liège, Belgium, 2002.

[7] C.Z. Janikow. Fuzzy Decision Trees: Issues and Methods. In *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, Volume 28, Number 1, pages 1-14, February, 1998.

[8] C. Marsala. Fuzzy Partitioning Using Mathematical Morphology in a Learning Scheme. In *Proceedings of the 5th International Conference on Fuzzy Systems, FUZZ-IEEE'96*. pages 1512-1517, New Orleans, September, 1996.

[9] C. Olaru and L. Wehenkel. A complete fuzzy decision tree technique. *Fuzzy Sets and Systems* 138 (2003) 221-254.

[10] C. Olaru. Contributions to Automatic Learning: Soft Decision Tree Induction. PhD Thesis, University of Liège, Belgium, 2003.

[11] A. Suarez and F. Lutsko. Globally Optimal Fuzzy Decision Trees for Classification and Regression. In *IEEE Transactions Pattern Analysis Machine Intelligence*. Volume 21, Number 12, pages 1297-1311, December, 1999.

[12] L. Wehenkel. Automatic learning techniques in power systems. Kluwer Academic, Boston, 1998.

[13] Y. Yuan and M.J. Shaw. Induction of fuzzy decision trees. *Fuzzy Sets and Systems* 69 (1995) 125-139.