

Keep the Decision Tree and Estimate the Class Probabilities Using its Decision Boundary

Isabelle Alvarez^(1,2)
(1) LIP6, Paris VI University
4 place Jussieu
75005 Paris, France
isabelle.alvarez@lip6.fr

Stephan Bernard⁽²⁾
(2) Cemagref, LISC
F-63172 Aubiere, France
stephan.bernard@cemagref.fr

Guillaume Deffuant⁽²⁾
(2) Cemagref, LISC
63172 Aubiere, France
guillaume.deffuant@cemagref.fr

Abstract

This paper proposes a new method to estimate the class membership probability of the cases classified by a Decision Tree. This method provides smooth class probabilities estimate, without any modification of the tree, when the data are numerical. It applies *a posteriori* and doesn't use additional training cases. It relies on the distance to the decision boundary induced by the decision tree. The distance is computed on the training sample. It is then used as an input for a very simple one-dimension kernel-based density estimator, which provides an estimate of the class membership probability. This geometric method gives good results even with pruned trees, so the intelligibility of the tree is fully preserved.

1 Introduction

Decision Tree (DT) algorithms are very popular and widely used for classification purpose, since they provide relatively easily an intelligible model of the data, contrary to other learning methods. Intelligibility is a very desirable property in artificial intelligence, considering the interactions with the end-user, all the more when the end-user is an expert. On the other hand, the end-user of a classification system needs additional information rather than just the output class, in order to assess the result: This information consists generally in confusion matrix, accuracy, specific error rates (like specificity, sensitivity, likelihood ratios, including costs, which are commonly used in diagnosis applications). In the context of decision aid system, the most valuable information is the class membership probability. Unfortunately, DT can only provide piecewise constant estimates of the class posterior probabilities, since all the cases classified by a leaf share the same posterior probabilities. Moreover, as a consequence of their main objective, which is to separate the different classes, the raw estimate at the leaf is highly biased. On the contrary, methods that are highly suitable for probability estimate produce generally less intelligible models. A lot of work aims at improving the class probability estimate at the leaf: Smoothing methods, specialized trees, combined methods (decision tree combined with other algorithms), fuzzy methods, ensemble methods (see section 2). Actually, most of these methods

(except smoothing) induce a drastic change in the fundamental properties of the tree: Either the structure of the tree as a model is modified, or its main objective, or its intelligibility.

The method we propose here aims at improving the class probability estimate without modifying the tree itself, in order to preserve its intelligibility and other use. Besides the attributes of the cases, we consider a new feature, the distance from the decision boundary induced by the DT (the boundary of the inverse image of the different class labels). We propose to use this new feature (which can be seen as the margin of the DT) to estimate the posterior probabilities, as we expect the class membership probability to be closely related to the distance from the decision boundary. It is the case for other geometric methods, like Support Vector Machines (SVM). A SVM defines a unique hyperplane in the feature space to classify the data (in the original input space the corresponding decision boundary can be very complex). The distance from this hyperplane can be used to estimate the posterior probabilities, see [Platt, 2000] for the details in the two-class problem. In the case of DT, the decision boundary consists in several pieces of hyperplanes instead of a unique hyperplane. We propose to compute the distance to this decision boundary for the training cases. Adapting an idea from [Smyth *et al.*, 1995], we then train a kernel-based density estimator (KDE), not on the attributes of the cases but on this single new feature.

The paper is organized as follows: Section 2 discusses related work on probability estimate for DT. Section 3 presents in detail the distance-based estimate of the posterior probabilities. Section 4 reports the experiment performed on the numerical databases of the UCI repository, the comparison between the distance-based method and smoothing methods. Section 5 discusses the use of geometrically defined subsets of the training set in order to enhance the probability estimate. We make further comments about the use of the distance in the concluding section.

2 Estimating Class Probabilities with a DT

Decision Trees (DT) posterior probabilities are piecewise constant over the leaves. They are also inaccurate. Thus they are of limited use (for ranking examples, or to evaluate the risk of the decision). This is the reason why a lot of work has been done to improve the accuracy of the posterior probabilities and to build better trees in this concern.

Traditional methods consist in smoothing the raw conditional probability estimate at the leaf $p^r(c|x) = \frac{k}{n}$, where k is the number of training cases of the class label classified by the leaf, and n is the total number of training cases classified by the leaf. The main smoothing method is the Laplace correction p^L (and its variants like m-correction). The correction $p^L(c|x) = \frac{k+1}{n+C}$, where C is the number of classes, shifts the probability toward the prior probability of the class ([Cestnik, 1990; Zadrozny and Elkan, 2001]). This improves the accuracy of the posterior probabilities and keep the tree structure unchanged, but the probabilities are still constant over the leaves. In order to bypass this problem, smoothing methods can be used on unpruned trees. The great number of leaves allows the tree to learn the posterior probability with more accuracy (see [Provost and Domingos, 2003]). The intelligibility of the model is most reduced in this case, so specialized building and pruning methods are developed for PET's (see for instance [Ferri and Hernandez, 2003]).

In order to produce smooth and case-variable estimate of the posterior probabilities, Kohavi [1996] deploys a Naive Bayes Classifier (NBC) at each leaf, using a specialized induction algorithm. Thus the partition of the space in NBTree is different from classical partition, since its objective is to better verify the conditional independence assumption at each leaf. In the same idea, Zang and Su [2004] use NBC to evaluate the choice of the attribute at each step. But the structure of the tree is essentially different.

Other methods obtain case-variable estimates of the posterior probabilities by propagating a case through several paths at each node (mainly fuzzy methods like in [Umano *et al.*, 1994] or in [Quinlan, 1993]; Or, more recently, [Ling and Yan, 2003]). These methods aim at managing the uncertainty both in the input and in the training database.

Smyth, Gray and Fayyad [1995] propose to keep the structure of the tree and to use a kernel-based estimate at each leaf. They consider all the training examples but use only the attributes involved in the path of the leaf. The dimension of the subspace is then at most the length of the path. But the resulting dimension is nevertheless far too high for this kind of technique and the method cannot be used in practice.

We propose here to reduce the dimension to 1, since KDE is very effective in very low dimensions. We preserve the structure of tree, and our method theoretically applies as soon as it is possible to define a metric on the input space.

3 Distance-based Probability Estimate for DT

3.1 Algebraic Distance as a New Feature

We consider an axis-parallel DT (ADT) operating on numerical data: Each test of the tree involves a unique attribute. We note Γ the decision boundary induced by the tree. Γ consists of several pieces of hyperplanes which are normal to the axes. We also assume that it is possible to define a metric on the input space, possibly with a cost or utility function.

Let x be a case, $c(x)$ the class label assigned to x by the tree, $d = d(x, \Gamma)$ the distance of x from the decision boundary Γ . The decision boundary Γ divides the input space into different areas (possibly not connected areas) which are labeled with the name of the class assigned by the tree.

By convention, in a two-class problem, we choose one class (the positive class) to orient Γ : If a case stands in the positive class area, its algebraic distance will be negative.

Definition 1 Algebraic distance to the DT (2-class problem)

The algebraic distance of x is $h(x) = -d(x, \Gamma)$ if $c(x)$ is the positive class c and $h(x) = +d(x, \Gamma)$ otherwise.

This definition extends easily to multi-class problems. For each class c , we consider Γ_c , the decision boundary induced by the tree for class c : Γ_c is the inverse image of the class c area. (We have $\Gamma_c \subset \Gamma$).

Definition 2 Class-Algebraic distance to the DT

The c -algebraic distance of x is $h_c(x) = -d(x, \Gamma_c)$ if $c(x) = c$ and $h_c(x) = +d(x, \Gamma_c)$ otherwise.

The c -algebraic distance measures the distance of a case to class c area. Actually, algebraic distance is a particular case of c -algebraic distance where c is the positive class.

These definitions apply to any decision tree. But in the case of axis-parallel DT (ADT) operating on numerical data, a very simple algorithm computes the algebraic distance (adapted from the distance algorithm in [Alvarez, 2004]). It consists in projecting the case x onto the set F of leaves f whose class label differs from $c(x)$ (in a two class problem). In a multi-class problem, each class c is considered in turn. When $c(x) = c$, the set F contains the leaves whose labels differs from c ; Otherwise it contains the leaves whose class label is c . The nearest projection gives the distance.

Algorithm 1 AlgebraicDistance(x, DT, c)

0. $d = \infty$;
1. Gather the set F of leaves f whose class $c(f)$ verifies:
($c(f) \text{ XOR } c(x) = c$);
2. For each $f \in F$ do: {
3. compute $p_f(x) = \text{projectionOntoLeaf}(x, f)$;
4. compute $d_f(x) = d(x, p_f(x))$;
5. if ($d_f(x) < d$) then $d = d_f(x)$ }
6. Return $h_c(x) = -\text{sign}(c(x) = c) * d$

Algorithm 2 projectionOntoLeaf($x, f = (T_i)_{i \in I}$)

1. $y = x$;
2. For $i = 1$ to $\text{size}(I)$ do: {
3. if y doesn't verify the test T_i then $y_u = b$ }
where T_i involves attribute u with threshold value b
4. Return y

The projection onto a leaf is straightforward in the case of ADT since the area classified by a leaf f is a hyper-rectangle defined by its tests. The complexity is in $O(Nn)$ in the worst case where N is the number of tests of the tree and n the number of different attributes of the tree.

The distance to the decision boundary presents two main advantages, because it is a continuous function: it is relatively robust to the uncertainty of the value of the attributes for new cases, and to the noise in the training data, assuming that only the thresholds of the tests are modified (not the attributes).

3.2 Kernel Density Estimate (KDE) on the Algebraic Distance

Kernel-based density estimation is a non-parametric technique widely used for density estimation. It is constantly im-

proved by researches on algorithms, on variable bandwidth and bandwidth selection (see [Venables and Ripley, 2002] for references).

Univariate KDE basically sums the contribution of each training case to the density via the kernel function K . To estimate the density $f(x)$ given a sample $M = \{x_i\}_{i \in [1, n]}$, one computes $\hat{f}(x) = \frac{1}{n} \sum_{i=1}^n \frac{1}{b} K\left(\frac{x-x_i}{b}\right)$, where K is the kernel function and b the bandwidth (b can vary over M).

Many methods could be used in this framework to compute the distance-based kernel probability estimate of the class membership. (We could also use kernel regression estimate). We have used very basic KDE for simplicity.

We consider here the algebraic distance $h(x)$ as the attribute on which the KDE is performed. So we compute the density estimate of the distribution of the algebraic distance, from the set of observations $h(x), x \in S$:

$$\hat{f}(h(x)) = \frac{1}{nb} \sum_{x_i \in S} K\left(\frac{h(x) - h(x_i)}{b}\right) \quad (1)$$

In order to estimate the conditional probabilities, we consider the set S of training cases and its subset S_c of cases such that $c(x) = c$. We estimate the density of the two populations:

\hat{f} the density estimate of the distribution of the algebraic distance to the decision boundary;

\hat{f}_c , the density estimate of the distribution of the algebraic distance of points of class c .

\hat{f} is computed on S and \hat{f}_c on S_c . We then derive from the Bayes rule, if $\hat{p}(c)$ estimates the prior probability of class c :

$$\hat{p}(c|h(x)) = \frac{\hat{f}_c(h(x)) * \hat{p}(c)}{\hat{f}(h(x))} \quad (2)$$

Definition 3 Distance-based kernel probability estimate

$\hat{p}(c|h(x))$ is called the distance-based kernel (DK) probability estimate

The algorithm is straightforward. We note S the training set used to build the decision tree DT .

Algorithm 3 DistanceBasedProbEst(x,DT,c,S)

1. Compute the algebraic distance
 $y = h(x) = \text{AlgebraicDistance}(x, DT, c)$;
2. Compute the subset $S(x)$ of S from which the probability density is estimated: Default value is $S(x) = S$;
3. Select $S_c(x) = \{x \in S(x), c(x) = c\}$;
4. Compute $\hat{f}(y) = \frac{1}{nb} \sum_{x_i \in S(x)} K\left(\frac{y-h(x_i)}{b}\right)$;
5. Compute $\hat{f}_c(y) = \frac{1}{nb} \sum_{x_i \in S_c} K\left(\frac{y-h(x_i)}{b}\right)$;
6. Compute and Return $\hat{p}(c|h(x))$ from equation (2)

Several possibilities can be considered for the set $S(x)$ used to compute the kernel density estimate, we discuss them in section 5. The simplest method consists in using the whole sample S . The algebraic distance is taken into account globally, without any other consideration concerning the location of the cases. We call the corresponding conditional probability estimate \hat{p}_g the global distance-based kernel (GDK) probability estimate. Figure 1 shows how the GDK probability estimate varies over a test sample, compared with the Laplace probability estimate which is piecewise constant.

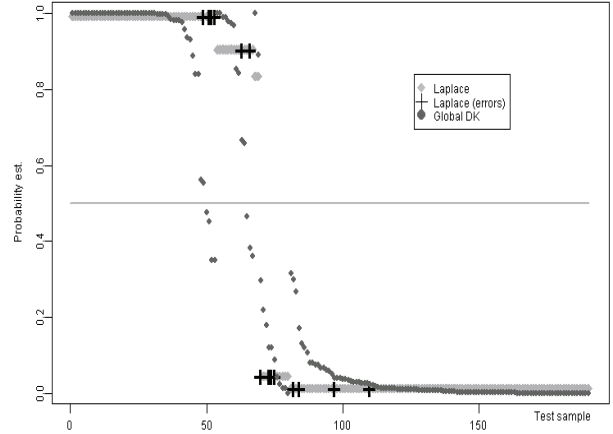


Figure 1: Variation of Laplace and GDK probability estimate over a test sample from the wdbc database. DT errors are highlighted

4 Experimental Study

4.1 Design and Remarks

We have studied the distance-based kernel probability estimate on the databases of the UCI repository [Blake and Merz, 1998] that have numerical attributes only and no missing values. For simplicity we have treated here multi-class problems as 2-class problems. We generally chose as positive class the class (or a group of class) with the lowest frequency in the database.

For each database, we divided 100 bootstrap samples into separate training and test sets in the proportion 2/3 1/3, respecting the prior of the classes (estimated by their frequency in the total database). It is certainly not the best way to build accurate trees for unbalanced datasets or different error costs, but here we are not interested in building the most accurate trees, we just want to study the distance-based kernel probability estimate. For the same reason we grow trees with the default options of j48 (Weka's [Witten and Frank, 2000] implementation of C4.5) although in many cases different options would build better trees. For unpruned trees we disabled the collapsing function. We used Laplace correction smoothing method to correct the raw probability estimate at the leaf for pruned trees and unpruned trees. We also built NBTREE on the same samples.

We used two different metrics in order to compute the distance from the decision boundary, the Min-Max (MM) metric and the standard (s) metric. Both metrics are defined with the basic information available on the data: An estimate of the range of each attribute i or an estimate of its mean E_i and of its standard deviation s_i . The new coordinate system is defined by (3).

$$y_i^{MM} = \frac{x_i - \text{Min}_i}{\text{Max}_i - \text{Min}_i} \quad \text{or} \quad y_i^s = \frac{x_i - E_i}{s_i} \quad (3)$$

The parameters of the standard metric are estimated on each training sample. (On each bootstrap sample for the Min-Max metric, to avoid multiple computation when an attribute is

outside the range). In practice, the choice of the metric should be guided by the data source. (For instance, the accuracy of most sensors is a function of the range, therefore it suggests to use an adapted version of the Min-Max coordinate system.)

To compute the kernel density estimate of the algebraic distance we choose simplicity and used the standard algorithm available in R [Venables and Ripley, 2002] with default options, although the setting of KDE parameters (choice of the kernel, optimal bandwidth, etc.) or specialized algorithms (also available in some dedicated packages) would certainly give better results. We systematically used the default option except for the bandwidth selection, since it is inappropriate for using the Bayes rule in equation (2): We used the same bandwidth for \hat{f} as for \hat{f}_c . We set it to a fraction τ of the range of the algebraic distance. This is the only parameter we used to control the KDE algorithm. More sophisticated methods could obviously be used but the Bayes rule in equation (2) should be reformulated if variable bandwidths are used.

4.2 Results for the Global Distance-based Kernel Probability Estimate

In order to compare our method with other methods of probability estimate, we used the AUC, the Area Under the Receiving Operator Characteristic (ROC) Curve (see [Bradley, 1997]). The AUC is widely used to assess the ranking ability of the tree. Although this method cannot distinguish between good scores and scores that are good probability estimates, it doesn't make any hypothesis on the true posterior probability, so it is useful when the true posterior probability are unknown (since it is clear that good probability estimates should produce on average better AUC). We also used Mean Squared Error and log-loss, although these methods make strong hypothesis on the true posterior probability.

Table 1 shows the difference of the AUC from global distance-based kernel (GDK) probability estimate and the Laplace correction. Apart from a few cases, it gives better values than Laplace correction (within a 95% confidence interval). From the intelligibility viewpoint, it is interesting to note that GDK probability estimate on pruned tree is generally better than smoothing method on unpruned tree (which is better than smoothing method on pruned tree). We also performed a Wilcoxon unilateral paired-wise test on each batch of 100 samples. The tests confirm exactly the significant results from Table 1 (with p-values always < 0.01).

The choice of the metric has a limited effect on the result in term of AUC. The difference is always of a different order except for vehicle, pima and especially glass for which it can reach 10^{-2} .

We used several bandwidths, with τ from 5 to 15% of the range of the algebraic distance: Results vary very smoothly with the bandwidth, Table 1 would present completely similar results.

Table 2 shows the comparison using Mean Squared-Error (MSE) as a metric to measure the quality of the probability estimate. The error at test point x is $s_e(x) = \sum_c (p(c|x) - \hat{p}(x|c))^2$, where $p(c|x)$ is the true conditional probability (here $p(c|x)$ is set to 1 when $c(x) = c$). $s_e(x)$ is then summed over each test sample. The performance of the GDK probability estimate diminishes quickly when τ increases, although

Dataset	Both	Both	Normal vs	DK vs.
	Red.-Error	Normal	Unpruned	NBTree
<i>bupa</i>	<i>0.26±0.77</i>	<i>0.39±0.84</i>	<i>0.03±0.81</i>	<i>0.91±0.93</i>
<i>glass</i>	<i>1.89±0.85</i>	<i>-0.0±0.7</i>	<i>-1.25±0.67</i>	<i>-1.02±0.78</i>
iono.	0.63±0.67	-2.2±0.53	-2.56±0.55	-3.02±0.64
<i>iris</i>	<i>4.70±0.52</i>	<i>3.85±0.46</i>	<i>3.90±0.48</i>	<i>1.99±0.47</i>
<i>thyroid</i>	<i>4.35±0.71</i>	<i>2.98±0.75</i>	<i>2.48±0.75</i>	<i>-0.57±0.75</i>
<i>pendig.</i>	<i>0.46±0.05</i>	<i>0.40±0.04</i>	<i>0.28±0.03</i>	<i>0.65±0.05</i>
<i>pima</i>	<i>2.78±0.58</i>	<i>0.31±0.41</i>	<i>0.41±0.44</i>	<i>0.69±0.54</i>
<i>sat</i>	<i>1.00±0.12</i>	<i>0.89±0.08</i>	<i>0.28±0.06</i>	<i>1.18±0.09</i>
<i>segment.</i>	<i>7.95±1.06</i>	<i>5.35±0.86</i>	<i>5.34±0.89</i>	<i>3.77±0.71</i>
<i>sonar</i>	<i>2.66±0.95</i>	<i>2.07±0.78</i>	<i>1.88±0.80</i>	-3.53±1.03
<i>vehicle</i>	<i>0.42±0.26</i>	<i>0.65±0.20</i>	<i>0.08±0.20</i>	<i>1.97±0.29</i>
<i>vowel</i>	<i>4.19±0.52</i>	<i>3.09±0.30</i>	<i>2.79±0.28</i>	<i>2.13±0.32</i>
<i>wdbc</i>	<i>3.75±0.28</i>	<i>2.24±0.23</i>	<i>2.17±0.23</i>	<i>2.20±0.21</i>
<i>wine</i>	<i>5.35±0.51</i>	<i>3.31±0.36</i>	<i>2.97±0.34</i>	<i>0.25±0.23</i>

Table 1: Mean difference of the AUC obtained with global DK probability estimate (standard metric, $\tau = 10\%$) and with Laplace correction. GDK on pruned tree versus Laplace on pruned or unpruned tree. (Mean values and standard deviations are $\times 100$. Insignificant values are italic. Bad results are bold)

the AUC remains better. This is easily comprehensible: when τ increases, the kernel estimate tends to erase sharp variations. As a consequence, probability estimate cannot reach the extremes (0 or 1) easily. The Log loss metric is not shown since it gives useless results (almost always infinite).

5 Local estimate: Partition of the Space

In order to get more local estimate, the first idea would be to use the leaves to refine the definition of the sets $S(x)$ used to run the kernel density estimate (step 2 of algorithm 3). $S(x)$ would be simply the leaf that classifies x . However, we'll argue on a simple example that this option shows severe drawbacks, and that a definition based on the geometry of the DT boundary should generally be preferred.

Dataset	GD vs. Laplace		GD vs. NBTree	
	MSE	p-value	MSE	p-value
<i>bupa</i>	<i>-7.29±0.59</i>	<i>3.9e-16</i>	<i>-1.68±0.88</i>	<i>0.006</i>
<i>glass</i>	<i>0.79±3.56</i>	0.017	<i>4.19±3.52</i>	<i>0.114</i>
iono.	1.88±0.27	2.2e-09	1.61±0.77	0.023
<i>iris</i>	<i>-1.54±0.37</i>	<i>5.3e-4</i>	<i>-0.84±0.65</i>	<i>0.088</i>
thyroid	1.49±0.71	<i>0.092</i>	6.04±0.82	1.3e-10
<i>pendig.</i>	<i>-0.08±0.01</i>	<i>1.9e-08</i>	<i>-0.19±0.03</i>	<i>2.0e-07</i>
<i>pima</i>	<i>-5.04±0.30</i>	<i>3.2e-18</i>	<i>-1.39±0.53</i>	<i>0.008</i>
<i>sat</i>	<i>-0.07±0.03</i>	<i>0.024</i>	<i>-0.20±0.05</i>	<i>5.0e-4</i>
<i>segment.</i>	<i>0.68±0.39</i>	0.017	<i>-1.46±0.76</i>	<i>0.071</i>
<i>sonar</i>	<i>-3.29±0.51</i>	<i>5.8e-09</i>	<i>0.34±1.52</i>	<i>0.491</i>
vehicle	1.99±0.50	5.7e-04	1.63±0.56	0.007
<i>vowel</i>	<i>-0.12±0.12</i>	<i>0.26</i>	0.69±0.19	1.1e-4
<i>wdbc</i>	<i>-2.63±0.22</i>	<i>3.8e-17</i>	<i>-2.00±0.40</i>	<i>4.6e-06</i>
<i>wine</i>	<i>-1.72±0.31</i>	<i>6.2e-08</i>	1.79±0.56	8.3e-4

Table 2: Mean MSE difference (and p-value of the corresponding unilateral Wilcoxon paired test) between GDK probability estimate (standard metric, $\tau = 5\%$) and either Laplace correction (normal pruned tree) or NBTree. (Values are $\times 100$ except p-values. Insignificant values are italic. Bad results for GDK are bold)

5.1 DT Boundary-based partition

The rationale behind our partition is to consider the groups of points for which the distance is defined by the same normal to the decision surface, because these points share the same axis defining their distance, and they relate to the same linear piece of the decision surface.

Let $\{H_1, H_2, \dots, H_k\}$ be the separators used in the nodes of the tree, and S the learning sample. To partition the total sample S , we associate with each separator H_i a subset S_i of S , which comprises the examples x such that the projection of x onto the decision boundary Γ belongs to H_i . Generically, the projection $p(x)$ of x onto Γ is unique, but $p(x)$ can belong to several separators. In that case we associate to x the separator H_i defining $p(x)$ with the largest margin. We define $S(x)$ as S_i . The KDE is then run on the $h(x_j), x_j \in S_i$, as explained previously.

5.2 Advantage of our partition

We now illustrate on a simple example the advantage of our partition compared with the partition generated by the leaves. Suppose that the learning examples are uniformly distributed in a square in a space x, y , and that the probability of class 1 depends only on axis x , with the following pattern (Figure 2):

- For x growing from -0.5 to 0 , the probability of class 1 grows smoothly from 0.1 to 0.9 , with a sigmoid function centered on -0.25 .
- For x growing from 0 to 0.5 , the probability of class 1 decreases sharply from 1 to 0 around $x = 0.25$.

A typical decision tree obtained from a sample generated with this distribution is represented on figure 2 (top). It has three leaves, two yield class -1 and are defined by a single separation, and one yields +1 and is defined by both separations (the gray area on the figure). The partition defined by the leaves thus generates 3 subsets.

Figure 2 (bottom) shows the partition obtained with our method. It includes only two subset, one for each separation. Figure 3 shows the results obtained with a kernel estimation (a simple Parzen window of width 0.1) on the leaf partition, and on our partition, compared with the actual probability function which was used to generate the sample. One can notice that our partition gives a result which is very close to the one that would be obtained with a kernel estimation on the x axis. This is not the case for the leaf partition, which introduces strong deviations from the original probability.

The main explanation of these deviations is a side effect which artificially increases the probability of 1 around the border, in the case of the leaf partition. Actually, this is the same bias as the one obtained when estimating directly the probabilities on the leaves. Moreover, the leaf partition introduces an artificial symmetry on both sides of the leaf, because of the averaging on all the examples at similar distance of the boundary. These problems are not met with our partition. We claim that the problems illustrated in this example have good chances to occur very frequently with the leaf partition. In particular, the side effect due to the border of the leaf will always induce a distortion, which is avoided with our partition. Artificial symmetries will always be introduced when a leaf includes several separators.

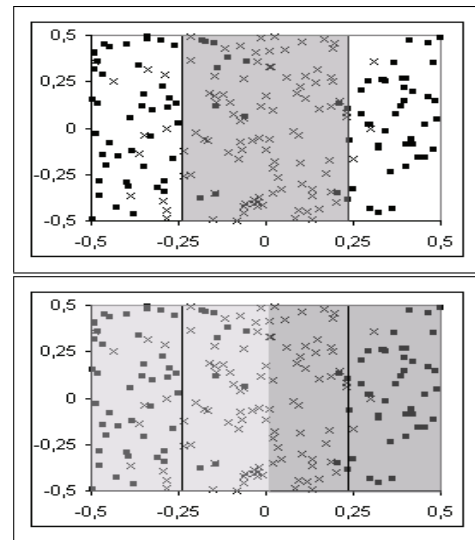


Figure 2: Top: Partition defined by the three leaves of the tree. Bottom: Partition into 2 subsets (light gray and gray), defined by the separators.

Table 3 shows the results of the comparison between local distance-based kernel estimate and smoothing method at the leaf for some databases. Because of the partition of the space, the results from the AUC comparison and MSE are not necessarily correlated, and MSE is globally better than for the global DK probability estimate.

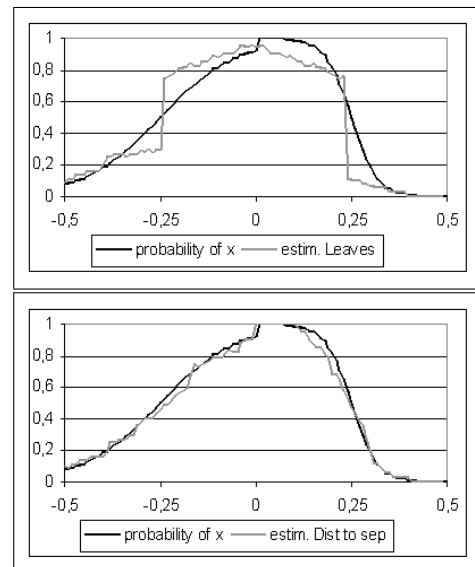


Figure 3: Results with a kernel estimator of width 0.1 (Parzen window). Top: With the leaf partition. Bottom: With the separator partition. (In black the true probability, in gray the estimate).

Dataset	Both Normal	Normal vs Unpruned	MSE (Normal)	p-value
bupa	-2.87±0.82	-3.23±0.80	-1.80±0.88	0.041
glass	-1.26±0.93	-2.42±0.90	0.38±0.4	0.090
iono.	-1.74±0.71	-2.07±0.74	-0.42±0.43	0.134
iris	4.31±0.52	4.37±0.55	-4.09±0.45	2.e-13
thyroid	1.60±0.78	1.09±0.78	0.45±0.59	0.335
pendigits	-1.65±1.09	-1.76±1.09	-0.10±0.01	5.e-8
pima	-3.00±0.64	-2.91±0.64	-2.76±0.51	3.e-06
sat	-0.73±0.74	-1.35±0.75	-0.13±0.03	4.e-06
sonar	1.06±0.79	0.87±0.80	-2.30±0.60	2.e-4
vehicle	-6.04±1.52	-6.61±1.51	2.02±0.45	9.e-07
vowel	0.94±0.91	0.63±0.89	0.00±0.15	0.537
wdbc	1.56±0.29	1.49±0.30	-2.15±0.28	1.e-10
wine	2.97±0.47	2.63±0.45	-1.63±0.45	5.e-06

Table 3: Comparison between local DKPE (standard metric, $\tau = 5\%$) and smoothing method: AUC mean difference and MSE. (All values expect p-values are $\times 100$. Insignificant values are italic. Bad results for Local DK are bold)

6 Conclusion

We have presented in this article a geometric method to estimate the class membership probability of the cases that are classified by a decision tree. It applies *a posteriori* to every axis-parallel tree with numerical attributes. The geometric method doesn't depend on the type of splitting or pruning criteria that is used to build the tree. It only depends on the shape of the decision boundary induced by the tree, and it can easily be used for real multi-class problem (with no particular class of interest). It consists in computing the distance to the decision boundary (that can be seen as the Euclidean margin). A kernel-based density estimator is trained on the same learning sample than the one used to build the tree, using only the distance to the decision boundary. It is then applied to provide the probability estimate. The experimentation was done with basic trees and very basic kernel estimate functions. But it shows that the geometric probability estimate performs well (the quality was measured with the AUC and the MSE).

We also proposed a more local probability estimate, based on a partition of the input space that relies on the decision boundary and not on the leaves boundaries.

The main limit of the method is that the attributes are numeric. It could be extended to ordered attributes, but it cannot be used with attributes that have unordered modalities. The methods could also be used with oblique trees, but the algorithms to compute the Euclidean distance are far less efficient. Further work is in progress in order to improve the local estimate. Another feature, linked to the nearest part of the decision boundary, could be used to train the kernel density estimator (which are still very efficient in dimension 2).

References

[Alvarez, 2004] I. Alvarez. Explaining the result of a decision tree to the end-user. In *Proc. of the 16th European Conf. on Artificial Intelligence*, pages 119–128, 2004.

[Blake and Merz, 1998] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[Bradley, 1997] A. P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30:1145–1159, 1997.

[Cestnik, 1990] B. Cestnik. Estimating probabilities: A crucial task in machine learning. In *Proc. of the European Conf. on Artificial Intelligence*, pages 147–149, 1990.

[Ferri and Hernandez, 2003] Flach P. Ferri, C. and J. Hernandez. Improving the auc of probabilistic estimation trees. In *Proc. of the 14th European Conf. on Machine Learning.*, pages 121–132, 2003.

[Kohavi, 1996] Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 202–207, 1996.

[Ling and Yan, 2003] C. X. Ling and R. J. Yan. Decision tree with better ranking. In *Proc. of the 20th Int. Conf. on Machine Learning*, pages 480–487, 2003.

[Platt, 2000] J. Platt. Probabilistic outputs for support vector machines. In Bartlett P. Schoelkopf B. Schurmans D. Smola, A.J., editor, *Advances in Large Margin Classifiers*, pages 61–74, Cambridge, 2000. MIT Press.

[Provost and Domingos, 2003] F. Provost and P. Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52(3):199–215, 2003.

[Quinlan, 1993] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, 1993.

[Smyth *et al.*, 1995] Padhraic Smyth, Alexander Gray, and Usama M. Fayyad. Retrofitting decision tree classifiers using kernel density estimation. In *Int. Conf. on Machine Learning*, pages 506–514, 1995.

[Umamo *et al.*, 1994] M. Umamo, K. Okomato, I. Hatono, H. Tamura, F. Kawachi, S. Umezu, and J. Kinoshita. Fuzzy decision trees by fuzzy id3 algorithm and its application to diagnosis systems. In *3rd IEEE International Conference on Fuzzy Systems*, pages 2113–2118, 1994.

[Venables and Ripley, 2002] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, fourth edition, 2002.

[Witten and Frank, 2000] I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementation*. Morgan Kaufmann, 2000.

[Zadrozny and Elkan, 2001] B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proc. 18th Int. Conf. on Machine Learning*, pages 609–616, 2001.

[Zhang and Su, 2004] H. Zhang and J. Su. Conditional independence trees. In *Proc. 15th European Conf. on Machine Learning*, pages 513–524, 2004.