Improving the AUC of Probabilistic Estimation Trees

César Ferri[1] Peter A. Flach[2] José Hernández-Orallo[1]

[1] Dep. Sistemes Informàtics i Computació, Univ. Politècnica de València, Spain
{cferri,jorallo}@dsic.upv.es
[2] Department of Computer Science, University of Bristol, United Kingdom
Peter.Flach@bristol.ac.uk

Abstract. In this work we investigate several issues in order to improve the performance of probabilistic estimation trees (PETs). First, we derive a new probability smoothing that takes into account the class distributions of all the nodes from the root to each leaf. Secondly, we introduce or adapt some new splitting criteria aimed at improving probability estimates rather than improving classification accuracy, and compare them with other accuracy-aimed splitting criteria. Thirdly, we analyse the effect of pruning methods and we choose a cardinality-based pruning, which is able to significantly reduce the size of the trees without degrading the quality of the estimates. The quality of probability estimates of these three issues is evaluated by the 1-vs-1 multi-class extension of the Area Under the ROC Curve (AUC) measure, which is becoming widespread for evaluating probability estimators, ranking of predictions in particular.

1. Introduction

Decision-tree learning has been extensively used in many application areas of machine learning, especially for classification, because the algorithms developed for learning decision trees [3,13] represent a good compromise between comprehensibility, accuracy and efficiency. In the common setting, a classifier is defined as a function from a set of m arguments or attributes (which can be either nominal or numeric) to a single nominal value, known as the class. We denote by C the set of c classes, usually simply referred by natural numbers 0, 1, 2, ... c-1. By E we denote the set of unlabelled examples. A classifier is a function f: $E \rightarrow C$. Traditionally, this setting was sufficient for most of the classification problems and applications. However, more and more applications require some kind of reliability, likelihood or numeric assessment of the quality of each classification. In other words, we do not only want that the model predicts a class value for each example but also that it can give an estimate of the reliability of each prediction. Such classifiers are usually called *soft classifiers*. Soft classifiers are useful in many scenarios, including combination of classifiers, cost-sensitive learning and safety-critical applications. The most general presentation of a soft classifier is a probability estimator, i.e. a model that estimates the probability $p_i(e)$ of membership of class $i \in C$ for every example $e \in E$.

A trained decision tree can be easily adapted to be a probability estimator by using the absolute class frequencies of each leaf of the tree. For instance, if a node has the following absolute frequencies n_1, n_2, ..., n_c (obtained from the training dataset) the estimated probabilities for that node can be derived as $p_i = n_i / \Sigma n_i$. Every new example falling into that leaf will have these estimated class probabilities. Such trees are

called *Probability Estimation Trees* (PETs). However, despite this simple conversion from a decision tree classifier into a PET, the probability estimates obtained by PETs are quite poor with respect to other probability estimators [14,2].

Some recent works have changed this situation. First, Provost and Domingos [12] improve the quality of PETs by reassessing some classical techniques in decision tree learning. In particular, they found that frequency smoothing of the leaf probability estimates, such as Laplace correction, significantly enhances the estimates, especially if they are used for ranking. On the other hand, pruning (or related techniques such as C4.5 collapsing) is shown to be unhelpful for increasing probability estimates. Unpruned trees usually give the best results. Independently, in an earlier paper [7] we also improve the quality of PETs by considering Laplace correction for the leaves. In addition, we showed that splitting criteria aimed at increasing accuracy (or reducing error), such as GainRatio, GINI or DKM [13,3,10] are not necessarily the best criteria for estimating good probabilities. Splitting criteria based on probability ranking, such as the new AUC-splitting criterion [7] can produce better results when the aim is to obtain good probability estimates. These two works are first steps that show that decision trees can be successfully used as probability estimators, provided we reassess and redefine some of the traditional techniques specifically devised for improving the accuracy of decision trees.

Provost and Domingos [12] "believe that a thorough study of what are the best methods for PETs would be a successful contribution to machine-learning research". In this spirit and as a sequel and natural continuation of the above-mentioned works, in this paper we present the following enhancements: (*i*) a new smoothing method (*m-branch smoothing*) for estimating probabilities, that not only considers the leaves, but all the frequencies from the root to the leaf; (*ii*) a new splitting criterion (*MSEEsplit*) defined in terms of the minimum squared error of the probability estimates; and (*iii*) a simple pruning criterion based on the cardinalities of nodes that is able to reduce the size of trees, without degrading the quality of the probability estimates.

The paper is organised as follows. In Section 2 we describe in more detail what a PET is and how it can be evaluated. Section 3 presents the new smoothing method. Section 4 introduces two new splitting criteria, MAUCsplit and MSEEsplit. Section 5 analyses the use of pruning and presents the influence of the degree of pruning of the best pruning method we have found so far for PETs. Finally, Section 6 discusses the results, and Section 7 closes the paper and proposes future work.

2. PETs, Features and Evaluation

In this section we present some necessary definitions, the evaluation framework, the experimental setting and some previous results in order to set the stage for the rest of the paper. The main contributions of this work are presented in subsequent sections.

Given the set of unlabelled examples E and the set C of c classes, we define a *probability estimator* as a set of c functions $p_{i \in C}: E \to \hat{A}$ such that $\forall p_{i \in C}, e \in E : 0 \leq p_i(e) \leq 1$ and $\forall e \in E \sum p_{i \in C}(e) = 1$. Decision trees are formed of nodes, splits and conditions. A *condition* is any Boolean function $g: E \to \{true, false\}$. A *split* is a set of s conditions $\{g_k : 1 \leq k \leq s\}$. In this paper, we consider the conditions of a split to be exhaustive and exclusive, i.e., for a given example one and only one of the conditions

of a split is true. A *decision tree* can be defined recursively as follows: (*i*) a node with no associated split is a decision tree, called a leaf; (*ii*) a node with an associated split $\{g_k : 1 \leq k \leq s\}$ and a set of s children $\{t_k\}$, such that each condition is associated with one and only one child, and each child t_k is a decision tree, is also a decision tree. Given a tree t there is just a single node r that is not child of any other node. This special node is called the *root* of the tree. The sequence of nodes $<\nu_1, \nu_2, ..., \nu_d>$ from the root to a leaf l, where $\nu_d = l$ and ν_1 is the root, is called the *branch* leading to l.

In the most straightforward and classical scenario a decision tree is learned by using a training set T, which is a set of labelled examples, i.e., a set of pairs of the form $<e, i>$ where $e \in E$ and $i \in C$. After the training stage, the examples will have been distributed among all the nodes in the tree, where the root node contains all the examples and downward nodes contain the subset of examples that are consistent with the conditions of the specific branch. Therefore, every node has particular absolute frequencies $n_1, n_2, ..., n_c$ for each class. The cardinality of the node is given by Σn_i. A *decision tree classifier* (DTC) is defined as a decision tree with an associated labelling of the leaves with classes. Usually, the assigned class is the most frequent class in the leaf ($argmax_i\{n_i\}$). A *probability estimation tree* (PET) is a decision tree where each leaf is assigned a probability distribution over classes. These probability estimates can for instance be relative frequencies $p_i = n_i / \Sigma n_i$.

2.1 DTCs, PETs and their Evaluation

One of the first questions that may arise is whether a good DTC is always a good PET and vice versa. Although there is a high correlation between quality of DTCs and quality of PETs, some recent works have shown that many heuristics used for improving classification accuracy "reduce the quality of probability estimates" [12]. Hence, it is worth investigating new heuristics and techniques which are specific to PETs and that may have been neglected by previous work in DTCs.

But first of all, a standard measure for evaluating the quality of PETs must be established. As justified and used by [12,7] and other previous work, the AUC (Area under the ROC Curve) measure has been chosen for evaluation. The measure can be interpreted as the probability that a randomly chosen example e of class 0 will have an estimated $p_0(e)$ greater than the estimated $p_1(e)$. Consequently, this is a measure particularly suitable for evaluating ranked two-class predictions. Recently, an extension of the AUC measure for more than two classes has been proposed by Hand and Till [9]. The idea is to simply average the AUC of each pair of classes (1-vs-1 multi-class). We call this measure *MAUC* for multi-class AUC (Hand and Till denote the function by M). Clearly, MAUC = AUC when c=2.

In [7] we introduced a new method for efficiently computing MAUC based on the ranking of leaves rather than a ranking of examples. Hence, the complexity of the new method depends on the number of leaves rather than on the number of examples, frequently entailing better performance. In what follows, we use this optimisation.

2.2 Datasets and Experimental Methodology

We evaluated the methods presented in this paper on 50 datasets from the UCI repository [1]. Half of them have two classes, either originally or by selecting one of the

classes and joining all the other classes, and the rest have more than two classes (multi-class datasets). The datasets are described in Table 1 and Table 2. The first two columns show the dataset number and name, the size (number of examples), the numbers of nominal and numerical attributes and the size of the minority class.

Table 1. Two-class datasets used.

#	Dataset	Size	Attributes		%Min
			Nom	Num	Class
1	Monks1	566	6	0	50
2	Monks2	601	6	0	34.28
3	Monks3	554	6	0	48.01
4	Tic-tac	958	8	0	34.66
5	House-votes	435	16	0	38.62
6	Agaricus	8124	22	0	48.2
7	Breast-wdbc	569	0	30	37.26
8	Breast-can-wisc	699	0	9	34.48
9	Breast-wpbc	194	0	33	23.71
10	Ionosphere	351	0	34	35.9
11	Liver-Bupa	345	0	6	42.03
12	Pima-Abalone	768	0	8	34.9
13	Chess-kr-vs-kp	3196	36	0	47.78
14	Sonar	208	0	60	46.63
15	Hepatitis	83	14	5	18.07
16	Thyroid-hypo	2012	19	6	6.06
17	Thyroid-sick-eu	2012	19	6	11.83
18	Yeast2c	1484	0	8	31.20
19	Spect	267	22	0	20.60
20	Habermn-Brst	306	0	3	26.47
21	Spam	4601	0	57	39.40
22	Cyl-Bands	365	19	17	36.99
23	Pima-Diabetes	768	0	8	34.90
24	Sick	2751	21	6	7.92
25	Lymph_2c	142	15	3	42.96

Table 2. Multi-class datasets used.

#	Dataset	#Classes	Size	Attributes		%Min
				Nom	Num	
26	Hypothyroid_3c	3	2750	21	6	3.24
27	Balance-Scale	3	625	0	4	7.84
28	Cars	4	1728	6	0	3.76
29	Dermatology	6	366	33	1	5.46
30	New-Thyroid	3	215	0	5	13.95
31	Nursery4C	4	12957	8	0	2.53
32	Page-Blocks	5	5473	0	10	0.51
33	Pendigits	10	10992	0	16	9.60
34	Tae	3	151	2	3	32.45
35	Iris	3	150	0	4	33.33
36	Optdigits	10	5620	0	64	9.86
37	Segmentation	7	2310	0	19	14.29
38	Wine	3	178	0	13	26.97
39	Heart-Dis-All	5	920	8	5	3.04
40	Anneal	5	898	32	6	0.89
41	Hayes-Roth	3	160	4	0	19.38
42	Waveform	3	5000	0	21	32.94
43	Cmc	3	1473	7	2	22.61
44	Ecoli4C	4	336	0	7	7.44
45	Autos-Drvwhls	3	205	9	16	4.39
46	Solar Flarec	3	323	10	0	2.17
47	HorseColicOutC	3	366	13	8	14.21
48	Ann-Thyroid	3	7200	15	6	2.31
49	Splice	3	3190	60	0	24.04
50	Sat	6	6435	0	36	9.73

All experiments have been done within the SMILES system (`http://www.dsic.upv.es/~flip/smiles/`). The use of the same system for all the methods makes comparisons more impartial because all other things remain equal. We used the basic configuration of the system, which is a decision-tree learner quite similar to C4.5, but without pruning (unless stated), without node "collapsing" [13], and the GainRatio splitting criterion used by default (this configuration is sometimes called C4.4).

We performed a 20 times 5-fold cross-validation, thus making a total of 50 x 100 = 5,000 runs of SMILES for each method. We have used 5-fold cross-validation instead of 10-fold cross-validation because for computing the AUC we need examples of all the classes and some datasets have a small proportion of examples for the minority class. In what follows, for each dataset we show the arithmetic mean and the standard deviation of the 100 runs. Accuracy and AUC are shown as a percentage.

2.3 Results with Laplace and *m*-Estimate Smoothing

Previously, we have stated that, given any node with absolute frequencies n_1, n_2, ..., n_c for each class (hence overall cardinality Σn_i), we can obtain a probability estimation tree by obtaining the probabilities as $p_i = n_i / \Sigma n_i$. One problem is that pure nodes with small cardinality will have the same probability as pure nodes with much higher cardinality. This is especially problematic for ranking predictions of unpruned trees,

because most or all nodes tend to be pure and there are many ties between the rankings. A common solution to this problem is the use of *probability smoothing* such as Laplace correction and *m*-estimate, defined as follows:

$$\textit{Laplace smoothing} \quad p_i = \frac{n_i + 1}{\left(\sum_{i \in C} n_i\right) + c} \qquad \textit{m-estimate smoothing} \quad p_i = \frac{n_i + m \cdot p}{\left(\sum_{i \in C} n_i\right) + m}$$

where c is the number of classes. The probability p in the m-estimate is the expected probability without any additional knowledge, and it is either assumed to be uniform ($p = 1/c$) or estimated from the training distribution. In the uniform case, which we used in our experiments, it is easy to see that Laplace correction is a special case of the m-estimate with $m=c$.

Table 3. Effect of smoothing on AUC for two-class datasets (m=4).

#	Without Smoothing		Laplace Smoothing		m-estimate Smoothing	
	AUC	SD	AUC	SD	AUC	SD
1	96.8	3.7	97.9	2.7	97.9	2.7
2	71.2	4.3	70.2	4.8	70.2	4.8
3	97.7	1.3	99.1	0.9	99.1	0.9
4	76.2	3.6	87.2	2.8	87.2	2.8
5	93.6	2.5	98.2	1.4	98.2	1.4
6	100.0	0.0	100.0	0.0	100.0	0.0
7	91.9	2.9	96.8	1.8	96.8	1.8
8	93.3	2.1	97.9	1.1	97.9	1.1
9	59.6	8.9	64.2	9.5	64.2	9.5
10	90.8	4.2	94.6	3.7	94.6	3.7
11	61.1	6.1	67.0	6.8	67.0	6.8
12	67.2	1.7	79.0	1.4	79.0	1.4
13	99.5	0.3	100.0	0.1	100.0	0.1
14	66.0	7.6	73.5	7.2	73.5	7.2
15	65.5	11.6	75.7	9.1	75.7	9.1
16	90.5	3.9	97.9	1.0	97.9	1.0
17	80.7	3.1	85.1	2.6	84.6	2.7
18	64.8	2.9	74.3	2.8	74.3	2.8
19	67.4	7.1	74.3	7.1	75.0	7.2
20	56.3	6.6	64.3	7.5	64.3	7.5
21	91.7	0.9	96.9	0.5	96.9	0.5
22	66.6	5.4	68.5	5.1	68.5	5.1
23	65.9	4.2	75.3	3.8	75.3	3.8
24	89.3	3.1	98.7	0.8	98.7	0.8
25	78.7	7.8	87.3	6.9	87.3	6.9
ARITM	79.3		85.0		85.0	
GEOM	78.0		83.9		84.0	

Table 4. Effect of smoothing on AUC for multi-class datasets (m=4).

#	Without Smoothing		Laplace Smoothing		m-estimate Smoothing	
	AUC	SD	AUC	SD	AUC	SD
26	97.9	1.4	99.8	0.3	99.8	0.3
27	75.0	1.8	82.9	2.7	82.9	2.7
28	94.7	2.1	95.4	1.6	95.4	1.6
29	98.4	0.8	99.0	0.6	99.0	0.6
30	94.3	3.5	97.1	2.7	97.1	2.7
31	99.4	0.3	99.7	0.1	99.7	0.1
32	94.4	1.8	97.8	1.0	97.8	1.0
33	99.3	0.1	99.7	0.0	99.7	0.0
34	75.0	7.9	74.7	8.5	74.7	8.5
35	97.3	2.2	98.5	1.8	98.5	1.8
36	98.2	0.2	99.0	0.1	99.0	0.1
37	99.3	0.2	99.7	0.1	99.7	0.1
38	96.6	2.7	97.8	1.9	97.8	1.9
39	63.7	3.7	65.6	3.6	65.6	3.6
40	98.6	2.0	99.2	1.1	99.2	1.1
41	89.8	4.2	90.5	4.3	90.5	4.3
42	83.4	1.0	88.8	0.9	88.8	0.9
43	62.0	2.6	65.5	2.7	65.5	2.7
44	93.0	2.9	95.3	2.5	95.3	2.5
45	88.1	8.4	93.0	5.8	93.0	5.8
46	57.5	8.1	58.6	10.4	58.6	10.4
47	66.3	5.4	70.5	4.9	70.5	4.9
48	98.1	1.0	99.8	0.2	99.8	0.2
49	95.6	0.6	98.1	0.4	98.1	0.4
50	95.1	0.3	96.9	0.3	96.9	0.3
ARITM	88.4		90.5		90.5	
GEOM	87.3		89.5		89.5	

Tables 3 and 4 show the results (mean and standard deviation for the 5 × 20 iterations) without smoothing, with Laplace smoothing and with the m-estimate with uniform prior (the best experimental value for m, m=4 is used). These results are similar to those of [12,7] and they are shown here to serve as a reference from which we will illustrate our own improvements. The improvement of Laplace and m-estimate smoothing over no smoothing is obvious — especially for two-class datasets — and there is no need to perform a significance test. On the other hand, there is virtually no difference between Laplace smoothing and the best m-estimate.

3. *m*-Branch Smoothing

We continue to investigate whether the previous results can be further improved. In this section we propose a more sophisticated smoothing method called *m-branch smoothing*. In the next section we consider alternative splitting criteria that are designed specifically for probability estimation trees.

First of all, the previous *m*-estimate and Laplace smoothing methods consider a uniform class distribution of the sample. That is, they consider the global population uniform whereas in many cases the class probabilities are unbalanced. However, just taking this into account does not improve the measures significantly, since each node takes a subsample from the upper node, and this, once again, makes a subsample of the upper node, until the root is reached. Usually, this means that the sample used to obtain the probability estimate in a leaf is the result of many sampling steps, as many as the depth of the leaf. It makes sense, then, to consider this *history* of samples when estimating the class probabilities in a leaf. The idea is to assign more weight to nodes that are closer to the leaf.

Definition 1 (*m*-Branch Smoothing). Given a leaf node l and its associated branch $<\nu_1, \nu_2, ..., \nu_d>$ where $\nu_d = l$ and ν_1 is the root, denote with n_i^j the cardinality of class i at node ν_j. Define $p_i^0 = 1/c$. We recursively compute the probabilities of the nodes from 1 to d as follows:

$$p_i^j = \frac{n_i^j + m \cdot p_i^{j-1}}{\left(\sum_{i \in C} n_i^j\right) + m}$$

The *m*-branch smoothed probabilities of leaf l are given by p_i^d.

We note that *m*-branch smoothing is a recursive root-to-leaf extension of the *m-probability estimate* used by Bratko and Cestnik for decision tree pruning [5].

Since this is an iteration of the *m*-estimate, we could use a fixed value of m. However, if we use a small m the smoothing would almost be irrelevant for upper nodes, which have high cardinality. On the other hand, if we use a large m the small cardinalities at the bottom of the branch would have low relevance. In order to solve this we use a variable value, which depends on the size of the dataset and the depth. Define the *height* of a node as $h = d+1-j$ where d is the depth of the branch and j the depth of the node. The normalised height of a node is defined as $\Delta = 1 - 1/h$ in order to increase the correction closer to the root. We then parametrise the m value as follows:

$$m = M \cdot \left(1 + \Delta \cdot \sqrt{N}\right)$$

where M is a constant and N is the global cardinality of the dataset. The use of the square root of N is inspired by "the square root law", which connects the error and the sample size. The previous expression means that *m*-branch smoothing is performed with a value of M at the leaves, the next node up is done with $M + \frac{1}{2} \cdot M \cdot \sqrt{N}$, the next $M + \frac{2}{3} \cdot M \cdot \sqrt{N}$ until the root with $M + \frac{(d-1)}{d} \cdot M \cdot \sqrt{N}$.

In Tables 5 and 6 we compare *m*-branch smoothing (with the best experimental value for M=4) compared with the best previous results (*m*-estimate). We also perform a paired *t*-test to test the significance of the results. The 'Better?' column indicates whether *m*-branch smoothing performs significantly better (✓) or worse (x) than

m-estimate smoothing, according to *t*-test with level of confidence 0.1. A tie (-) indicates the difference is not significant at this level.

Table 5. Comparison of *m*-estimate and *m*-branch smoothing on two-class datasets.

#	M-ESTIMATE SMOOTHING		M-BRANCH SMOOTHING		BETTER?
	AUC	SD	AUC	SD	
1	97.9	2.7	97.2	3.4	×
2	70.2	4.8	67.4	5.0	×
3	99.1	0.9	99.1	1.0	-
4	87.2	2.8	86.9	2.7	-
5	98.2	1.4	98.5	1.4	-
6	100.0	0.0	100.0	0.0	-
7	96.8	1.8	96.9	1.6	-
8	97.9	1.1	98.0	1.1	-
9	64.2	9.5	65.9	10.0	-
10	94.6	3.7	94.4	3.7	-
11	67.0	6.8	70.0	7.1	✓
12	79.0	1.4	82.2	1.4	✓
13	100.0	0.1	99.9	0.1	×
14	73.5	7.2	75.7	6.3	✓
15	75.7	9.1	77.5	9.4	-
16	97.9	1.0	98.1	1.0	-
17	84.6	2.7	86.1	2.7	✓
18	74.3	2.8	75.7	2.6	✓
19	75.0	7.2	77.9	7.0	✓
20	64.3	7.5	67.3	7.0	✓
21	96.9	0.5	97.0	0.5	-
22	68.5	5.1	68.5	5.2	-
23	75.3	3.8	78.8	3.3	✓
24	98.7	0.8	98.7	0.8	-
25	87.3	6.9	87.4	6.9	-
ARITMEAN	85.0		85.8		8 wins, 14 ties,
GEOMEAN	84.0		84.9		3 losses

Table 6. Comparison of *m*-estimate and *m*-branch smoothing on multi-class datasets.

#	4-ESTIMATE SMOOTHING		4-BRANCH SMOOTHING		BETTER?
	AUC	SD	AUC	SD	
26	99.8	0.3	99.8	0.2	✓
27	82.9	2.7	81.3	2.9	×
28	95.4	1.6	95.3	1.5	-
29	99.0	0.6	99.2	0.5	✓
30	97.1	2.7	97.4	2.7	-
31	99.7	0.1	99.7	0.1	×
32	97.8	1.0	98.8	0.7	✓
33	99.7	0.0	99.8	0.0	✓
34	74.7	8.5	75.0	8.7	-
35	98.5	1.8	98.5	1.8	-
36	99.0	0.1	99.3	0.1	✓
37	99.7	0.1	99.7	0.1	✓
38	97.8	1.9	97.8	1.8	-
39	65.6	3.6	69.1	3.6	✓
40	99.2	1.1	98.6	2.2	×
41	90.5	4.3	91.7	4.2	✓
42	88.8	0.9	95.0	0.5	✓
43	65.5	2.7	71.1	2.4	✓
44	95.3	2.5	95.6	2.6	-
45	93.0	5.8	91.7	7.1	-
46	58.6	10.4	59.3	12.0	-
47	70.5	4.9	76.2	5.2	✓
48	99.8	0.2	99.8	0.3	-
49	98.1	0.4	98.7	0.3	✓
50	96.9	0.3	98.3	0.2	✓
ARITMEAN	90.5		91.5		13 wins, 9 ties,
GEOMEAN	89.5		90.6		3 losses

The results (21 wins, 23 ties, 6 losses) show that there are many cases where the difference is not significant (especially when the AUC was close to 100) but there are many more cases where the results are improved than degraded. In overall geometric means, *m*-branch smoothing improves AUC with 1% from 86.7% to 87.7%.

4. Splitting Criteria for PETs

A crucial factor for the quality of a decision tree learner is its splitting criterion. A variety of splitting criteria, including Gini [3], Gain, Gain Ratio and C4.5 criterion [13], and DKM [10] have been presented to date. However, all these were designed and evaluated for classifiers, not for probability estimators. In this section we propose and investigate two splitting criteria specifically designed for PETs.

4.1 MAUC Splitting Criterion

In [7] we introduced a novel splitting criterion, which was aimed at maximising the AUC of the resulting tree rather than its accuracy. It simply computes the quality of each split as the AUC of the nodes resulting from that split, assuming a two-class

problem. This can be generalised to more than two classes using Hand and Till's 1-vs-1 average [9].

Definition 2 (MAUCsplit). Given a split s, the quality of the split is defined as:

$$MAUCsplit(s) = MAUC(t_s)$$

where t_s indicates the tree with the node being split as root.

The idea of using the same measure for splitting that is used as well for evaluation seems straightforward. Nonetheless, in the same way that accuracy (expected error) is not necessarily the best splitting criterion for accuracy, MAUCsplit may not the best splitting criterion for MAUC.

4.2 MSEE Splitting Criterion

A different approach is to consider that the tree really *predicts probabilities*. It thus makes sense to minimise the quadratic error committed when guessing these probabilities. Consider a split where each of the children has estimated probabilities p_i for each class. Assume that nodes assign classes according to p_i. Consequently, p_i means the probability of examples of class i falling into the node but also means the probability of being classified as i. Assuming these two interpretations of p_i are independent, the probability that an example of class i is misclassified, denoted by $p_{e,i}$, can be estimated as follows:

$$p_{e,i} = p_i \cdot \sum_{j \neq i} p_j = p_i \cdot (1 - p_i)$$

In words, this combines the probability that an example is of class p_i and the probability that it is not classified accordingly (the sum of the rest of probabilities, which is 1 − p_i). This is similar to the Gini index. However, we want to measure the quadratic error of the *prediction*, which in our case is not a class but a probability. Hence, given a misclassification:

- p_i should have been 1 but is p_i. Thus, the error can be estimated as $(1 - p_i)^2$.
- p_j ($j \neq i$) should have been 0 and is p_j. The error can be estimated as $(0 - p_j)^2$.

Consequently, we have a total quadratic error of:

$$Error_i = p_i \cdot (1 - p_i)\left((1 - p_i)^2 + \sum_{j \neq i} (0 - p_j)^2\right) = p_i \cdot (1 - p_i)\left((1 - p_i)^2 + \sum_{j \neq i} p_j^{\,2}\right)$$

Therefore, if we consider a split of n nodes, then we can compute the quality of the split as the negative value of the total error for all the nodes:

Definition 3 (MSEEsplit). Given a split s, the quality of the split is defined as:

$$MSEEsplit(s) = \sum_{k=1..n} q_k \cdot \left(-\sum_{i=1..c} Error_i\right)$$

where q_k indicates the relative cardinality of the k-th child in the split.

The way in which the error is obtained gives the name for the criterion: *Minimum Squared Expected Error* (MSEE). Note that this expression is similar to the Brier score [4], which has also been used recently as a measure for predictive models in similar applications as where AUC is used.

Both MAUCsplit and MSEEsplit are modified in order to penalise splits with a high number of children, in a similar way as GainRatio is a modification of the Gain criterion. The precise correction we have used in the experiments can be found in [8].

4.3 Splitting Criteria Comparison

We have compared several splitting criteria: GainRatio (as implemented in C4.5, i.e., considering only the splits with Gain greater than the mean [13]), MGINI (as implemented in CART [3]), DKM (as presented in [10]), MAUCsplit with children correction and MSEEsplit with children correction. We will show the results with the split smoothing that gives better results for each criterion. This smoothing has not to be confused with the smoothing used for computing the AUC for evaluating the PETs, which will always be *m*-branch smoothing. Table 7 summarises the results (the complete results can be found in [8]).

Table 7. Summary of Accuracy and AUC for several splitting criteria (geometric means).

		C4.5SPLIT	GAIN	MGINI	DKM	MAUCSPLIT	MSEE SPLIT	BETTER? MSEE VS C4.5
2-CLASS	ACCURACY	81.4	81.6	81.4	81.7	81.8	82.0	11 wins, 9 ties, 5 losses
	AUC	84.9	84.8	84.6	84.8	85.0	85.3	7 wins, 13 ties, 5 losses
>2-CLASS	ACCURACY	82.8	83.0	83.1	83.1	82.4	83.0	10 wins, 11 ties, 4 losses
	AUC	90.6	90.9	90.8	91.1	90.8	90.9	7 wins, 13 ties, 5 losses
ALL	ACCURACY	82.1	82.3	82.2	82.4	82.1	82.5	21 wins, 20 ties, 9 losses
	AUC	87.7	87.8	87.7	87.9	87.8	88.1	14 wins, 26 ties, 10 losses

According to these and previous results, the best DTC splitting criterion is DKM, but the difference is not significant with the rest of DTC criteria (MGINI, C4.5). The new criterion MAUCsplit is slightly better than C4.5 and MGINI, although differences are small and not significant. Finally, MSEEsplit appears to be the best, although differences are smaller with respect to C4.5 and even smaller with respect to DKM. The good behaviour of both MSEE and DKM may be explained because both methods use quadratic terms.

5. Pruning and PETs

As we have discussed in the introduction, in [12] it is argued that pruning is counterproductive for obtaining good PETs and, consequently, pruning (and related techniques) should be disabled. However, it is not clear whether the reason is that pruning is intrinsically detrimental for probability estimation, or that existing pruning methods are devised for accuracy and not for increasing AUC.

Independently, we have evaluated some classical pre-pruning and post-pruning methods, such as Expected Error Pruning and Pessimistic Error Pruning (see e.g. [6] for a comparison). Our results match those of [12]; even slight pruning degrades the quality (measured in terms of AUC) of the probability estimates. It seems that smoothing has a relevant effect here: if we disable smoothing, pruning is beneficial in some cases. Consequently, it looks as though the better the smoothing at the leaves is, the worse pruning will be. It appears that this will be especially true for our *m*-branch smoothing, since it takes into account all the branch nodes probabilities. Pruning will

reduce the available information for estimating the probabilities. As a result, we do not expect to obtain new pruning methods that will increase the AUC of a PET, but we might be interested in designing pruning methods that reduce the size of the tree without degrading too much the quality of the PET.

One of the most important issues for estimating good probabilities is the size of the sample. Consequently, the poorest estimates of a PET will be obtained by the smallest nodes. If we have to decide to prune some nodes it makes sense to prune the smallest ones first. This would suggest a very simple pre-pruning method: nodes will not be expanded when their cardinality is lower than a certain constant. However, datasets with a large number of classes can have poor probability estimates with medium-large nodes if there are many small classes. Hence, we can refine cardinality-based pruning, by using the following definition:

Definition 4 (CardPerClass Pruning). Given a node l, it will be pruned when:

$$Card(l) < 2\frac{K}{c}$$

where $Card(l)$ is the cardinality of node l, K is a constant (K=0 means no pruning) and c is the number of classes.

In the following graph, we show the effect of CardPerClass pruning (with K-values ranging from 16 to 0). The results are shown for MSEEsplit with m-branch smoothing.

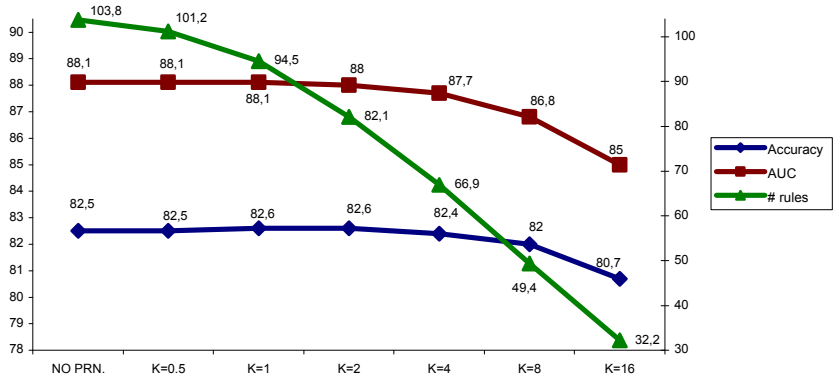

Fig. 1. Accuracy, AUC and number of rules for several pruning degrees (geometric mean).

As can be seen in Figure 1, only strong pruning is counterproductive for accuracy (and even behaves worse than other pruning methods). It is more interesting to observe the evolution of the AUC curve. The graph suggests that the quality of a PET is not significantly decreased until K=4, which, on the other hand, leads to a considerable decrease in the complexity of the trees.

6. Summary

In previous sections we have presented several enhancements in order to improve the AUC of PETs. In order to see the whole picture, we show below the accumulated progress of the techniques presented before.

Although there is a considerable improvement obtainable by using a simple smoothing such as Laplace smoothing (as shown previously [12,7]), there was still place for further improvement, as can be seen in Table 8. According to the nature and number of the datasets, and the quantity and quality of work developed for improving decision trees, we think that this is a significant result.

Table 8. Summary Table of AUC (only AUC and geomeans shown).

	C4.5SPLIT WITHOUT SMOOTH		C4.5SPLIT WITH LAPLACE SMOOTH		C4.5SPLIT WITH MBRANCH SMOOTH		MSEESPLIT WITH MBRANCH SMOOTH		C4.5LAP VS MSEESPLIT MBRANCH + K=1 PRUNING
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	BETTER IN AUC?
2-CLASS	81.4	78.0	81.4	83.9	81.4	84.9	82.1	85.4	11 wins, 9 ties, 5 losses
>2-CLASS	82.8	87.3	82.8	89.5	82.8	90.6	83.1	90.9	16 wins, 4 ties, 5 losses
ALL	82.1	82.5	82.1	86.7	82.1	87.7	82.6	88.1	27 wins, 13 ties, 10 losses

7. Conclusions and Future Work

In this work we have reassessed the construction of PETs, evaluating and introducing new methods for the three issues that are most important in PET construction: leaf smoothing, splitting criteria and pruning. We have introduced a new *m*-branch smoothing method that takes the whole branch of decisions into account, as well as a new MSEE splitting criterion aimed at reducing the squared error of the probability estimate.

Our new *m*-branch smoothing is significantly better than previous classical smoothings (Laplace or *m*-estimate). With respect to the splitting criteria, there are few works that compare existing splitting criteria for accuracy. Moreover, to our knowledge, this is the first work that compares the ranking of probability estimates of several splitting criteria for PETs. At this point, the conclusion is that all the good criteria presented so far are also good criteria for AUC and the differences between them are negligible. Nonetheless, pursuing new measures, we have found new splitting criteria such as AUCsplit and MSEEsplit comparable to the best known criterion (or even better, although this is not conclusive). Finally, we have shown that a simple cardinality pruning method can be applied (to a certain extent) to obtain simpler PETs without degrading their quality too much. Consequently, the idea that pruning is intrinsically bad for PETs is still in question, or, at least, we reiterate that a statement of its negative influence is "inconclusive" [12]. A very recent work has also suggested that a mild pruning could be beneficial [11].

As future work, other methods for improving the estimates (without modifying the structure of a single tree) such as the method presented in [11] (which uses the frequencies of all the leaves on the trees) could yield a method that takes into account all the information in the tree. Additionally, we think that better pruning methods for PETs could still be developed (considering the size of the dataset as an additional

factor) — these might include the use of the *m-branch estimate* for pruning (as similar measures were originally introduced [5]).

Acknowledgments

This work has been partially supported by CICYT grant TIC2001-2705-C03-01. We would also like to thank the referees for their useful suggestions and references.

References

1. Blake, C., Merz, C. UCI repository of machine learning databases, University of California (http://www.ics.uci.edu/~mlearn/MLRepository.html), 1998.
2. Bradley, A.P. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7): 1145-1159, 1997.
3. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J. *Classification and regression trees*. Belmont, CA, Wadsworth, 1984.
4. Brier, G.W. Verification of forecasts expressed in terms of probability. *Monthly Weather Rev.*, 78: 1-3, 1950.
5. Cestnik, B., Bratko, I. On estimating probabilities in tree pruning. In *Proc. European Working Sessions on Learning* (EWSL-91), Lecture Notes in Artificial Intelligence 482, pp.138-150, Springer-Verlag, 1991.
6. Esposito, F., Malerba, D., Semeraro, G. A Comparative Analysis of Methods for Pruning Decision Trees. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(5): 476-491, 1997.
7. Ferri, C., Flach, P., Hernández-Orallo, J. Learning Decision Trees using the Area Under the ROC Curve. In C. Sammut; A. Hoffman (eds.), *Proc. Int. Conf. on Machine Learning (ICML2002)*, pp. 139-146, Morgan Kaufmann, 2002.
8. Ferri, C., Flach, P., Hernández-Orallo, J.. *Decision Trees for Ranking: Effect of new smoothing methods, new splitting criteria and simple pruning methods.* Tech. Rep. Dep. de Sistemes Informàtics i Computació, Univ. Politècnica de València, 2003.
9. Hand, D.J., Till, R.J. A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems. *Machine Learning*, 45: 171-186, 2001.
10. Kearns, M., Mansour, Y. On the boosting ability of top-down decision tree learning algorithms. *Journal of Computer and Systems Sciences,* 58(1): 109-128, 1999.
11. Ling, C.X., Yan, R.J. Decision Tree with Better Ranking. In *Proc. Int. Conf. on Machine Learning (ICML2003)*, AAAI Press, 2003.
12. Provost, F., Domingos, P. Tree Induction for Probability-based Ranking. *Machine Learning* 52(3), 2003.
13. Quinlan, J.R. *C4.5. Programs for Machine Learning*. San Francisco, Morgan Kaufmann, 1993.
14. Smyth, P., Gray, A., Fayyad, U. Retrofitting decision tree classifiers using kernel density estimation. In *Proc. Int. Conf. on Machine Learning (ICML1995)*, pp. 506-514, Morgan Kaufmann, 1995.