Machine Learning manuscript No.
(will be inserted by the editor)

A Comparison of Pruning Criteria for Probability Trees

Daan Fierens, Jan Ramon, Hendrik Blockeel, Maurice Bruynooghe

K.U.Leuven, Dept. of Computer Science, Celestijnenlaan 200A, B-3001 Leuven, e-mail: {Daan.Fierens, Jan.Ramon, Hendrik.Blockeel, Maurice.Bruynooghe}@cs.kuleuven.be

Received: date / Revised version: date

Abstract Probability trees are decision trees that predict class probabilities rather than the most likely class. The pruning criterion used to learn a probability tree strongly influences the size of the tree and thereby also the quality of its probability estimates. While the effect of pruning criteria on classification accuracy is well-studied, only recently there is more interest in the effect on probability estimates. Hence, it is currently unclear which pruning criteria for probability trees are preferable under which circumstances.

In this paper we survey six of the most important pruning criteria for probability trees, and discuss their theoretical advantages and disadvantages. We also perform an extensive experimental study of the relative performance of these pruning criteria. The main conclusion is that overall a pruning criterion based on randomization tests performs best because it is most robust to extreme data characteristics (such as class skew or a high number of classes). We also identify and explain several shortcomings of the other pruning criteria.

Key words Decision Trees, Pruning, Probability Estimation, Randomization Tests

1 Introduction

Probability trees (sometimes called Probability Estimation Trees or PET's) are decision trees with probability distributions on a set of classes in the leaves (Provost & Domingos, 2003). Probability trees are useful in several ways. First, they can be used directly for *probability estimation*, for instance as a compact way of specifying conditional probability distributions in graphical models (Friedman & Goldszmidt, 1998; Heckerman, Chickering, Meek, Rounthwaite, & Kadie, 2000). Second, they can be used for *ranking* instances according to the probability of belonging to a certain class (probability-based ranking (Provost & Domingos, 2003)).

Probability trees, and decision trees in general, are usually learned by first constructing a large tree and then pruning this tree. Pruning is done by traversing the tree in a bottom-up manner while removing subtrees that are not beneficial according to some pruning criterion.

1.1 Goal

The goal of this paper is to gain insight into the performance of some of the most important pruning criteria for probability trees. The motivation for this goal is the fact that the degree of pruning used when learning a probability tree greatly influences the quality of the resulting probability estimates.

- If we prune too much, the resulting probability tree will be too small and will not perform well because its probability estimates will be too unrefined. Concretely, some instances will be given the same probability estimates (because the instances are grouped in the same leaf of the tree), while it would be better to give them different, more refined probability estimates (by further splitting the tree such that the instances are in separate leaves).
- If we do not prune sufficiently, the resulting probability tree will be too large and might not perform well because its probability estimates are badly calibrated. Roughly speaking, calibration is the degree to which probability estimates correspond to fractions in the data (Zadrozny & Elkan, 2001). Probability estimates become better calibrated if they are estimated from more examples. For a given dataset size, larger trees have fewer examples in each leaf and are hence more susceptible to poor calibration.

The above shows that pruning is a key issue when learning probability trees. The effect of pruning criteria on classification accuracy has been well-studied for more than a decade (Esposito, Malerba, & Semeraro, 1997; Frank & Witten, 1998). However, it has been argued that to obtain good probability estimates one might have to use other pruning criteria than to obtain good classification accuracy (Provost & Domingos, 2003). Studies on the effect of pruning criteria on probability estimates or probability-based rankings appeared only recently. For instance, Provost and Domingos (2003) compare error based pruning with not using pruning, Neville et al. (2003) compare randomization test pruning with chi-square pruning, and Ferri et al. (2003) introduce cardinality-based pruning. However, none of the above papers compare more than two pruning criteria at the same time. Hence, there is no clear view on the relative performance of all the different criteria. Concretely, it is unclear which pruning criteria for probability trees are preferable under which circumstances (in terms of dataset characteristics and performance measures). The goal of this paper is to gain more insight into this.

1.2 Contributions

The main contributions of this paper are as follows. First, we give a survey of six pruning criteria, including a discussion of their theoretical advantages and disadvantages. Second, we perform an extensive experimental study of their relative

performance, and we identify and explain the shortcomings of each pruning criterion.

This paper is a continuation of previous work in which we experimentally compared a number of pruning criteria (Fierens, Ramon, Blockeel, & Bruynooghe, 2005). This paper significantly extends our previous work in three ways. First, this paper includes randomization tests as a new pruning criterion. Since this criterion performs best in our experiments, this is a significant addition. Second, the experimental comparison is now carried out on 26 datasets instead of 12, increasing the confidence in the conclusions. Third, whereas in our previous work we used area under the ROC curve (AUC) as the main performance measure, we now also include other measures that directly evaluate the probability estimates themselves rather than the probability-based rankings as AUC does (concretely, we include squared error and conditional likelihood).

1.3 Structure of the Paper

The remainder of this paper is structured as follows. In Section 2 we discuss the basics of learning probability trees. In Section 3 we discuss six of the most important pruning criteria for probability trees. In Section 4 we experimentally compare these pruning criteria. In Section 5 we discuss some related pruning criteria that have been proposed in other work. In Section 6 we discuss the computational complexity of the pruning criteria. Finally, in Section 7 we conclude.

2 Learning Probability Trees

In this section we give some background on learning probability trees which is needed for the rest of the paper. First we give a generic algorithm for learning decision trees using pruning. Then we show how this generic algorithm can be used for learning probability trees. Finally, we give some details about the tree learner that we used in this paper.

We only consider binary trees in this paper. The motivation for this choice is discussed in Section 2.3.

2.1 Learning Decision Trees Using Pruning

Decision trees are usually learned by first constructing a large ‘unpruned’ tree in a top-down manner and then pruning this tree in a bottom-up manner.[1]

[1] This is called *postpruning*. It is also possible to apply *prepruning*: learn a tree top-down and use a pruning criterion to decide when to stop the construction of the tree along a branch. In earlier work we experimentally compared prepruning and postpruning (Fierens, Ramon, Blockeel, & Bruynooghe, 2007). Our results showed that postpruning performs better than prepruning, at the cost of an increased running time. This is consistent with the common knowledge about pruning for classification trees.

procedure LEARN_UNPRUNED_TREE
in: $\mathcal{D}$: dataset, $\mathcal{C}$: set of candidate-tests
out: tree

if $\mathcal{D}$ is pure
then
  return CREATE_LEAF($\mathcal{D}$)
else
  for each candidate-test $T \in \mathcal{C}$
    $h(T) =$SPLITTING_HEUR$(T, \mathcal{D})$
  end for
  $T_{best} = argmax_{T \in \mathcal{C}}(h(T))$
  $\mathcal{D}_{left} = \{e \in \mathcal{D} | T_{best}$ succeeds for $e\}$
  $\mathcal{D}_{right} = \mathcal{D} \setminus \mathcal{D}_{left}$
  if $\mathcal{D}_{left} \neq \emptyset$ **and** $\mathcal{D}_{right} \neq \emptyset$
  then
    $\mathcal{T}_{left} =$LEARN_UNPRUNED_TREE$(\mathcal{D}_{left}, \mathcal{C})$
    $\mathcal{T}_{right} =$LEARN_UNPRUNED_TREE$(\mathcal{D}_{right}, \mathcal{C})$
    return $tree(\mathcal{T}_{left}, T_{best}, \mathcal{T}_{right})$
  else
    return CREATE_LEAF($\mathcal{D}$)

Fig. 1 Algorithm for learning an unpruned decision tree.

To learn an unpruned tree, we are given a dataset of labelled examples and a set of candidate tests. We start from the empty tree. For each candidate test we compute the heuristic value according to a splitting criterion (for instance information gain). We then look for the candidate test T_{best} with the highest heuristic value. If this test effectively splits the dataset (succeeds for some examples and fails for others) we add it to the tree and we continue growing the tree by applying the same procedure recursively to learn the left and right subtrees. Otherwise we return a leaf. The corresponding algorithm is shown in Figure 1.

Pruning a tree can be done in two ways which we call subtree pruning and substump pruning (a substump is a subtree with only one internal node). In *subtree pruning* we first recursively prune the left and right subtree, resulting in a new tree $\mathcal{T}_{new}$. Then we use a pruning criterion to decide whether $\mathcal{T}_{new}$ is better than a tree $\mathcal{T}_{leaf}$ consisting of a single leaf. If so we return $\mathcal{T}_{new}$, otherwise we return $\mathcal{T}_{leaf}$. *Substump pruning* is very similar to subtree pruning. The only difference is that we only consider the possibility of replacing a tree $\mathcal{T}_{new}$ by a leaf if $\mathcal{T}_{new}$ is a stump, but not if it is larger. The algorithms for subtree and substump pruning are shown in Figure 2, the difference between the two algorithms is indicated by a box.

The advantage of subtree pruning is that it can remove a subtree of arbitrary size in a single step while substump pruning needs multiple steps to do this since it never removes more than a stump in each step. The disadvantage of subtree pruning is that it usually requires a more complex pruning criterion since deciding whether an arbitrary tree is better than a leaf is more complex than deciding whether a stump is better than a leaf.

procedure PRUNE
in: $\mathcal{T}$: unpruned tree, $\mathcal{D}$: dataset
out: pruned tree

if $\mathcal{T}$ is a leaf
then
  return $\mathcal{T}$
else
  // option1: prune the subtrees:
  decompose $\mathcal{T}$ in $\mathcal{T}_{left}$, T_{root} and $\mathcal{T}_{right}$
  $\mathcal{D}_{left} = \{e \in \mathcal{D} | T_{root}$ succeeds for $e\}$
  $\mathcal{D}_{right} = \mathcal{D} \setminus \mathcal{D}_{left}$
  $\mathcal{T}_{left\_new}$ =PRUNE$(\mathcal{T}_{left}, \mathcal{D}_{left})$
  $\mathcal{T}_{right\_new}$ =PRUNE$(\mathcal{T}_{right}, \mathcal{D}_{right})$
  $\mathcal{T}_{new} = tree(\mathcal{T}_{left\_new}, T_{root}, \mathcal{T}_{right\_new})$
  // option2: collapse into a leaf:
  $\mathcal{T}_{leaf}$ =COLLAPSE_TO_LEAF$(\mathcal{T}_{new})$
  // choose the best of the two options:
  if $\boxed{\mathcal{T}_{new} \text{ is a stump } \textbf{and}}$
    PRUNING_CRITERION$(\mathcal{T}_{new}, \mathcal{T}_{leaf}, \mathcal{D}) = true$
  then
    return $\mathcal{T}_{leaf}$
  else
    return $\mathcal{T}_{new}$

Fig. 2 Algorithm for subtree pruning (without box) and substump pruning (with box).

2.2 Parameters of the Algorithm for Learning Probability Trees

The algorithms of Figure 1 and 2 are the generic algorithms for learning and pruning decision trees. These algorithms contain three parameters that need to be instantiated in order to obtain a concrete algorithm for learning probability trees.

- The splitting criterion (used in the SPLITTING_HEUR function). In this paper we use information gain, gain ratio and Gini index (we perform all experiments with each of these three criteria) but any other splitting criterion could be incorporated as well.
- The pruning criterion. We discuss this in detail in Section 3.
- The approach for creating a leaf node (the CREATE_LEAF function), which amounts to estimating the class probabilities in the leaf. It is well-known that it is crucial to use smoothing when estimating class probabilities (Provost & Domingos, 2003; Ferri, Flach, & Hernandez-Orallo, 2003b). This is because non-smoothed probability estimates tend to be too extreme (too close to or even equal to 0 or 1), resulting in poor performance on unseen examples. We use Laplace correction for smoothing since it has been found very effective for probability trees despite being very simple and parameterless (Provost & Domingos, 2003; Ferri, Flach, & Hernandez-Orallo, 2003b). With Laplace correction, the estimated probability of class i in a leaf l is $\frac{N_l^i+1}{N_l+C}$ with N_l^i the number of

examples of class i in leaf l, N_l the total number of examples in leaf l, and C the number of classes. More complex smoothing techniques have been proposed as well, such as m-branch (Ferri, Flach, & Hernandez-Orallo, 2003b), curtailment (Zadrozny & Elkan, 2001), shrinkage (Wang & Zhang, 2006) and others (Ling & Yan, 2003), but they typically require tuning parameters.

2.3 Learning Probability Trees using Tilde

In our work we are interested in learning not only from *attribute-value datasets* (such as those in the UCI repository (Merz & Murphy, 1996)) where each instance is described by a fixed vector of attributes, but also from *relational datasets* (Džeroski & Lavrač, 2001) where instances have a more complex, relational structure such as molecules or citation networks (Kramer, De Raedt, & Helma, 2001; McCallum, Nigam, Rennie, & Seymore, 1999). In relational learning, probabilistic approaches such as upgrades of Bayesian networks currently attract a lot of attention (Getoor, Friedman, Koller, & Pfeffer, 2001; Kersting & De Raedt, 2007). As a consequence, probability estimation is currently an important problem in relational learning. Hence, when doing a study of probability trees, we believe it is important to include also relational datasets.

Because we want to be able to handle relational datasets, we implemented the algorithms for learning probability trees in the relational decision tree system Tilde (Blockeel & De Raedt, 1998; Van Assche, Vens, Blockeel, & Džeroski, 2006). Tilde was developed as a relational version of the well-known C4.5 system (Quinlan, 1993). As a representation for an instance Tilde uses a first-order logic interpretation, essentially a set of Prolog facts. Tests in internal nodes of a tree are Prolog queries (conjunctions of literals, possibly with aggregates). Since such tests either succeed or fail, trees learned by Tilde are always binary.

Tilde can of course also handle attribute-value datasets. When Tilde is applied on attribute-value data and with its standard settings, it coincides with C4.5 except for one difference: Tilde learns only binary trees whereas C4.5 can learn trees with also n-ary splits ($n>2$). When dealing with binary attributes, this does not make any difference. To deal with an n-ary attribute ($n>2$) in Tilde, we use n binary candidate tests. For example, if we have a ternary attribute that can have the value a, b or c, then we use three binary candidate tests in Tilde: testing whether the attribute equals a or not, b or not, and c or not. To deal with continuous attributes, we use minimal entropy discretization (Dougherty, Kohavi, & Sahami, 1995) to generate seven thresholds, and then use candidate tests of the form ‘attribute value is smaller than threshold’.

3 Pruning Criteria for Probability Trees

In this section we survey six pruning criteria for probability trees. To the best of our knowledge this is the first detailed survey of pruning criteria for probability trees.

When selecting the pruning criteria to be included in this paper, we focused on criteria that have been used before for probability trees and that are relatively well-known. Together the criteria included in this paper offer a good coverage of the different families of pruning criteria. First, from the information-theoretic or complexity-based criteria we include Minimum Description Length and Bayesian Information Criterion (there are several others such as Minimum Message Length (Wallace & Patrick, 1993) and Akaike Information Criterion (Domingos, 1998)). Second, from the criteria based on classification accuracy we include Error Based Pruning (there are many others, see Esposito et al. (1997)). Third, from the criteria using hypothesis tests for pairwise correlations we include chi-square tests and randomization tests. Finally we also consider the option of not pruning at all since this has been suggested as a good approach for learning probability trees (Provost & Domingos, 2003).

For each of the above six pruning criteria we discuss the main idea, the exact formulation, and the advantages and disadvantages. Note that we use some of the pruning criteria for substump pruning and others for subtree pruning. In general, we use subtree pruning when possible. This is the case for criteria that assign global scores to trees (such as Minimum Description Length). However, some criteria are of a local nature (such as criteria based on hypothesis tests for pairwise correlations) and cannot easily be used for subtree pruning, hence we use them for substump pruning.

3.1 Simple Hypothesis Tests (Chi-Square Test)

3.1.1 Main Idea For **substump pruning** we basically need a way to decide whether a stump should be retained or replaced by a leaf. One possible way to do this is to check whether the test in the internal node of the stump is really correlated with the class variable. This can be checked using any standard hypothesis tests such as the χ^2 (chi-square) -test. If the test in the stump is correlated with the class variable, then we retain the stump, otherwise we replace it by a leaf.

3.1.2 Details A naive approach could be to directly use a χ^2-test with a standard significance level ($\alpha = 0.05$) to check whether the test T in the internal node of the stump is correlated with the class variable or not. To do this we formulate the null hypothesis that T is not correlated with the class variable, and check whether we can reject this null hypothesis. Concretely, we compute the χ^2-score of T, and then compute the probability that a χ^2-score at least that high is obtained if the null hypothesis is true. If this probability is lower than 0.05, then we reject the null hypothesis and conclude that T is indeed correlated with the class variable.

The above naive approach suffers from the *'multiple tests problem'* or 'multiple comparisons problem' (Jensen & Cohen, 2000; Cohen & Jensen, 1997). The problem is that this approach does not take into account that T is not a random test but was chosen as the best test of a set of candidate tests. Not taking this into account increases the probability of a type I error, i.e., the probability that T will appear correlated with the class variable while it is not. Hence, we use *Bonferroni*

correction: instead of using a fixed significance level ($\alpha = 0.05$), we adapt the significance level to the number of candidate tests: $\alpha = \frac{0.05}{N_{test}}$ with N_{test} the number of candidate tests. This solves the multiple tests problem (Jensen & Cohen, 2000).

Hypothesis tests with Bonferroni correction can be used for substump pruning as follows. In the first phase we learn an unpruned tree but while doing so we check for each node whether the best test in that node is really correlated with the class variable, and we store this result for later use. In the second phase we then use substump pruning where we replace a stump by a leaf if the test in that stump was uncorrelated, and retain the stump otherwise. This pruning criterion has been proposed before by Jensen and Schmill (1997) . We refer to this pruning criterion as CHI.

3.1.3 Discussion This criterion has some disadvantages that arise from the assumptions typically made by hypothesis tests. There are three main assumptions. First, the Bonferroni correction for computing the significance level assumes that all candidate tests are mutually independent. When this is not the case the Bonferroni correction 'overcorrects' the significance level, thereby increasing the probability of wrongly rejecting the best test (Jensen & Cohen, 2000). As a consequence, trees might be learned that are too small to perform well. Second, to determine the probability of a certain χ^2-score, the chi-square sampling distribution is used. This is problematic because this sampling distribution is only appropriate if we have enough examples for each class (at least 5 to 10), which is often not the case when applying the χ^2-test in nodes towards the bottom of a tree, or when the data is strongly skewed (Frank & Witten, 1998). Third, it is assumed that examples are independent, which might be violated in some datasets (for instance, relational datasets). Since this last assumption is made by all pruning criteria in this paper, we do not discuss it further.

3.2 Randomization Tests

3.2.1 Main Idea Randomization tests are an alternative to simple hypothesis tests such as the chi-square test. The advantage is that they avoid most assumptions made by the chi-square test (and similar tests). Randomization tests can be used for **substump pruning** in the same way as chi-square tests in the previous section.

3.2.2 Details We again formulate the null hypothesis that the best test T_{best} of a set of candidate tests is not correlated with the class variable. To see whether we can reject this null hypothesis, we compute the heuristic value $h(T_{best})$ of T_{best}, and compute the probability of obtaining a heuristic value at least that high under the null hypothesis. To compute this probability using randomization tests, we construct an *empirical sampling distribution* rather than using a fixed parametric sampling distribution such as the chi-square distribution.

To construct an empirical sampling distribution we repeat the following procedure R times (we use $R = 100$): randomly permute the class labels of all examples

(thereby enforcing the null hypothesis), compute the heuristic value of each candidate test based on the class labels, select the best test T_{best}^{rand} and store its heuristic value $h(T_{best}^{rand})$. The probability of obtaining a heuristic value of at least $h(T_{best})$ is then estimated as the fraction of times that $h(T_{best}^{rand})$ is at least $h(T_{best})$. If this probability is lower than 0.05, then we reject the null hypothesis and conclude that T_{best} is indeed correlated with the class variable. Note that in the above procedure we already took into account the fact that T_{best} is not a random test but was chosen as the best test of a set of candidate tests. This avoids the multiple tests problem, making Bonferroni correction unnecessary.

As far as we know, this substump pruning strategy has not been used before. However, randomization tests have been used before for subtree pruning (Oates & Jensen, 1998) and for prepruning (Neville, Jensen, Friedland, & Hay, 2003; Frank & Witten, 1998). We discuss these existing approaches and the differences with our approach in Section 5. We refer to our approach as RAND.

3.2.3 Discussion The advantage of RAND is that it is very flexible: it avoids the first two assumptions made by the CHI criterion. Hence, it is no problem if candidate tests are mutually dependent, or if only a few examples are available of a certain class. A disadvantage of RAND is that it is computationally more expensive since for each candidate test a heuristic value needs to be computed R times instead of only once like for CHI. However, in practice this does not imply that RAND is a factor R slower than CHI, as we show in Section 6.

3.3 Minimum Description Length (MDL)

3.3.1 Main Idea In **subtree pruning** we need a way to decide whether a given tree is better than the same tree with some subtree replaced by a leaf. If we have a scoring criterion for trees, then we can simply compute the score for both trees and select the tree with the best score. Minimum Description Length (MDL) provides one way of defining such a scoring criterion.

3.3.2 Details MDL is a general criterion for determining the quality of a model. MDL tries to find a balance between the fit of the model to the training data and the complexity of the model. We use the MDL criterion for probability trees developed by Friedman and Goldszmidt (1998).[2]

The score of a probability tree $\mathcal{T}$ on a dataset $\mathcal{D}$ is the description length of the tree plus the description length of the data given the tree (lower scores are better).

$$score(\mathcal{T}, \mathcal{D}) = DL(\mathcal{T}) + DL(\mathcal{D} \mid \mathcal{T}) \quad (1)$$

[2] Other MDL criteria have been proposed for classification trees (Quinlan & Rivest, 1989; Mehta, Rissanen, & Agrawal, 1995) but these are not appropriate for probability trees since they rely on the fact that leaves contain a single class label, which leads to different complexity penalties.

Let N denote the total number of examples in the dataset, C the number of classes, and N_{test} the number of candidate tests in a node. The description length of the tree is defined as

$$DL(\mathcal{T}) = \begin{cases} 1 + 0.5(C-1)log_2(N) & \text{if } \mathcal{T} \text{ is a leaf} \\ 1 + log_2(N_{test}) + DL(\mathcal{T}_{left}) + DL(\mathcal{T}_{right}) & \text{otherwise,} \end{cases} \quad (2)$$

with $\mathcal{T}_{left}$ and $\mathcal{T}_{right}$ the subtrees of $\mathcal{T}$. This description length essentially serves as a complexity penalty for the tree. The description length of the data $\mathcal{D}$ given the tree $\mathcal{T}$ is defined as

$$DL(\mathcal{D} \mid \mathcal{T}) = \sum_{i=1}^{N} -log_2(p_{\mathcal{T}}(c_i|e_i)), \quad (3)$$

with c_i the true class label of example e_i, and $p_{\mathcal{T}}(c_i|e_i)$ the probability of this class label as predicted by the tree. This description length indicates how well the tree fits the data.

3.3.3 Discussion A disadvantage of MDL in general is that if often overpenalizes models: the complexity penalties are so high that a very simple model is selected, resulting in poor performance (Domingos, 1998). This has for instance been observed for classification trees (Quinlan & Rivest, 1989) and Bayesian networks (Chickering & Heckerman, 1997).

3.4 Bayesian Information Criterion (BIC)

3.4.1 Main Idea The Bayesian Information Criterion (BIC) is a general criterion for determining the quality of a model. BIC was originally proposed as a Bayesian approach to model selection (Schwarz, 1978), but it can also be interpreted as a reformulation of MDL (Friedman & Goldszmidt, 1998; Hastie, Tibshirani, & Friedman, 2001, Section 7.8). Since we already discussed MDL, we will focus on the latter interpretation. BIC can be used for **subtree pruning** in the same way as MDL in the previous section.

3.4.2 Details BIC can be seen as MDL with a modified definition of the description length of a model. In MDL the description length of a model depends on the entire model ("model structure" and number of independent numerical parameters), while in BIC it only depends on the numerical parameters.[3] As a consequence, the complexity penalties for BIC are smaller than those for MDL.

The BIC criterion in itself is well-known (Domingos, 1998) and is widely applied to certain kinds of models such as Bayesian networks (Chickering & Heckerman, 1997). However, it appears that we are the first to apply BIC to probability trees. We define the description length of a tree $\mathcal{T}$ as follows.

$$DL(\mathcal{T}) = \begin{cases} 0.5(C-1)log_2(N) & \text{if } \mathcal{T} \text{ is a leaf} \\ DL(\mathcal{T}_{left}) + DL(\mathcal{T}_{right}) & \text{otherwise} \end{cases} \quad (4)$$

[3] For probability trees, the parameters are the class probabilities in the leaves.

The description length of the data given the model is the same as for MDL (Equation 3).

3.4.3 Discussion Because complexity penalties for BIC are smaller than those for MDL, models learned by BIC are more complex than those learned by MDL (for probability trees this means that trees for BIC are larger than trees for MDL). This is usually an advantage since MDL often learns models that are too simple to perform well. Nevertheless, also BIC often learns models that are too simple because complexity penalties are still too high (Hastie et al., 2001). This has for instance been observed for Bayesian networks (Chickering & Heckerman, 1997).

3.5 Error Based Pruning (EBP)

3.5.1 Main Idea Error Based Pruning (EBP) was developed for classification trees. The idea behind EBP is to minimize the estimated number of errors that a tree would make on unseen data (Quinlan, 1993). The estimated number of errors of a tree is computed as the sum of the estimated number of errors of all its leaves. EBP can be used for **subtree pruning** as follows: we replace a subtree by a leaf if the estimated number of errors of the leaf is smaller than the estimated number of errors of the subtree.

3.5.2 Details To estimate the number of errors on unseen data that a leaf l in a tree would cause, the training data in l is considered as a statistical sample. The number of errors on unseen data (of the same size as the training data) for l is then estimated as the upper bound of a confidence interval around the actual number of errors on the training data in l. To be precise, let N_l denote the total number of examples in leaf l, and let E_l denote the actual number of errors on the training data in leaf l (i.e., the number of training examples in l that do not have the majority class of l). The error rate e_l on the training data in l is defined as the fraction $\frac{E_l}{N_l}$. Assuming that errors on the training data are binomially distributed, we can compute a confidence interval around this error rate. As in the C4.5 system we use a confidence level of 0.25 (Quinlan, 1993, Section 4.3). The upper bound u_l of this confidence interval is the estimated error rate on unseen data. The estimated number of errors of l is then the product $u_l N_l$.

EBP is one of the most widely used pruning strategies (Esposito et al., 1997). It is also the pruning strategy used by the C4.5 system (Quinlan, 1993).[4] There is one difference between EBP as explained above and EBP as implemented by C4.5: we do not apply 'grafting'. In grafting, a subtree can be replaced not only by a leaf, but also by one of its own subtrees (Quinlan, 1993; Esposito et al., 1997). The reason why we do not apply it for EBP is that other pruning criteria such as MDL and BIC have been proposed without grafting as well (although grafting is in principle possible), and we do not want to bias the comparison between EBP and these criteria.

[4] Next to EBP, C4.5 applies an additional form of pruning known as collapsing, even when building so called 'unpruned' trees (Provost & Domingos, 2003). Since collapsing harms probability estimates, we never apply it.

3.5.3 Discussion EBP tries to minimize the classification error, or in other words, to maximize the classification accuracy. It has been observed that on datasets with a very skewed class distribution EBP sometimes learns a very small tree that almost always predicts the majority class (Bradley, 1997; Zadrozny & Elkan, 2001). While this might indeed give good classification accuracy, this usually gives poor probability estimates. Hence, for probability trees EBP is expected not to be the best choice. Nevertheless, EBP is widely used, also for probability trees (Provost & Domingos, 2003; Ferri, Flach, & Hernandez-Orallo, 2003b), and hence we include it in our comparison.

3.6 No Pruning

3.6.1 Main Idea The last pruning criterion that we consider corresponds to not pruning at all: we simply learn an unpruned tree. Provost and Domingos (2003) incorporated this approach together with Laplace correction in the C4.5 system and called the result C4.4. In this paper, we refer to this approach as NOPRUNING.

3.6.2 Discussion Unpruned trees are typically very large (hundreds or even thousand of nodes on common datasets). Provost and Domingos (2003) argue that such trees are suited for probability estimation since they give a very fine-grained fragmentation of the instance space which makes it possible to accurately model any probability distribution on this space. However, they also mention a competing effect: such large trees typically have only a few examples in the leaves, which can result in overfitting (essentially due to poor calibration, recall Section 1.1). Another disadvantage of unpruned trees is that they are less interpretable since they are so large.

3.7 Relations Between the Six Pruning Criteria

For a given dataset, the trees learned by the above six pruning criteria all "overlap", i.e., the trees are all the same except for being less or more deep along some branches. This is because, apart from the pruning criterion, all other parameters of the learning algorithm (the splitting criterion, ...) are kept constant.

Figure 3 summarizes some relations that hold between the trees learned by the different pruning criteria. Obviously, the tree for NOPRUNING is the largest (this is the unpruned tree). Also, the tree for BIC is at least as large as the tree for MDL since the complexity penalty for BIC is lower than for MDL (recall Section 3.4.3).

4 Experimental Comparison of Pruning Criteria

In this section we perform a detailed experimental comparison of the six pruning criteria presented in the previous section.

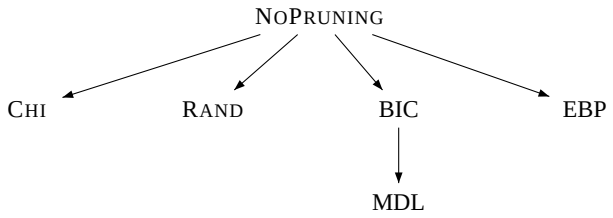

Fig. 3 Relations between the trees learned using the different pruning criteria (on the same dataset). A directed arc from a criterion C_1 to a criterion C_2 denotes that the tree for C_1 is at least as large as the tree for C_2.

4.1 Performance Measures

Performance measures for probabilistic classifiers can roughly be divided into two different groups: measures that evaluate the probability estimates themselves, and measures that depend only on the ranking or ordering determined by the probability estimates (as in probability-based ranking (Provost & Domingos, 2003)). For both groups, several alternative performance measures exist. It is known that measures from the same group are strongly correlated (Caruana & Niculescu-Mizil, 2004). Hence, we simply choose one well-known representative from each of the two groups.

From the first group of measures (the probability measures) we include the **root mean squared error (RMSE)**. The RMSE of a tree $\mathcal{T}$ on a dataset is defined as

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(1-p_{\mathcal{T}}(c_i|e_i))^2}, \tag{5}$$

where c_i denotes the true class label of example e_i, $p_{\mathcal{T}}(c_i|e_i)$ denotes the probability of this class label as predicted by the tree $\mathcal{T}$, and N denotes the total number of examples.

From the second group of measures (the ranking measures) we include **area under the ROC curve (AUC)**. For two-class datasets, AUC is the probability that a random positive example and a random negative example are ranked correctly (i.e., the positive example is predicted as more likely to be positive than the negative example). For multi-class datasets, we follow the common approach of computing a weighted average of the AUC's obtained by using in turn each of the classes as positive and all others as negative, where the weight of a class is the fraction of examples of that class in the data (Ferri, Hernández-Orallo, & Salido, 2003; Fawcett, 2001).

As performance measures, AUC and RMSE are complementary. RMSE depends on the probability estimates themselves. Hence, RMSE is quite sensitive to how well the probability estimates are calibrated. In contrast, AUC depends only on the ranking determined by the probability estimates. Changes to the probability estimates that do not modify the resulting ranking do not affect AUC. Hence, AUC is typically much less sensitive than RMSE to the calibration of the probability estimates. Concretely, poor calibration typically leads to poor performance

Table 1 Characteristics of the datasets: number of examples, number of classes, class skew (fraction of examples having the minority class; only for two-class datasets) and number of candidate-tests for the root node. The top shows the 20 attribute-value datasets, the bottom the 6 relational datasets.

	Examples	Classes	Skew	Tests
annealing	812	5		51
australian credit	653	2	45.3%	79
balance scale	625	3		16
breast wisconsin	683	2	35.0%	63
chess kr-kp	3196	2	47.8%	74
diabetes	768	2	34.9%	56
german credit	1000	2	30.0%	91
heart cleveland	296	2	46.0%	57
ionosphere	351	2	35.9%	225
mushroom	8124	2	48.2%	119
pen digits	7494	10		112
primary tumor	336	21		31
segment	2310	7		117
soybean	630	15		44
splice	3190	3		287
thyroid	3247	4		67
vehicle	846	4		126
voting	435	2	38.6%	48
vowel	990	11		87
yeast	1484	10		46
biodegradability	328	2	43.6%	569
cora	865	10		71
diterpenes	1503	23		186
gene (kdd'01)	1230	10		1761
hiv (ncs)	41768	2	3.6%	58
mutagenesis	230	2	40.0%	139

in terms of RMSE, while performance in terms of AUC can still be good (Caruana & Niculescu-Mizil, 2004).

4.2 Datasets and Experimental Setup

We use 20 attribute-value datasets and 6 relational datasets. All attribute-value datasets are standard benchmarks from the UCI repository (Merz & Murphy, 1996). All relational datasets are standard benchmarks from the fields of inductive logic programming or relational data mining. Table 1 gives the main characteristics of the datasets, more details can be found in Appendix A.1.

In addition to AUC and RMSE we also report tree size.[5] We report these results for all pruning criteria on all datasets. All reported results are averages over ten

[5] We measure tree size as the number of internal nodes of the tree, this equals the number of leaves minus one since we use binary trees.

different runs of five-fold cross-validation. All experiments were performed using three splitting criteria. Below, we show the results obtained with information gain as the splitting criterion. The results with gain ratio and Gini index are very similar, see Section 4.10.

For comparing two pruning criteria, we report *significant wins/ties/losses* according to two-tailed paired t-tests: a criterion wins if it is significantly better according to the t-test (with $p = 0.05$). To compute the t-tests we use the 'averaging over sorted runs' approach of Bouckaert (2004) since this avoids the usual problems with applying t-tests on repeated experiments, such as an elevated type I error and low replicability. In addition to significant wins/ties/losses we also report *'direct' wins/ties/losses* which do not involve any significance test: a criterion wins if its average result (over the runs of the cross-validation) is better.

4.3 Main Results

Tables 2, 3 and 4 give the results for RMSE, AUC and tree size. In these tables, we use the pruning criterion RAND as a reference against which we compute wins/ties/losses (because overall RAND performs best). The symbol "•" next to the result for a pruning criterion on a dataset means that this criterion is significantly better than RAND on this dataset (significant win). The symbol "◦" means that the criterion is significantly worse than RAND (significant loss). In addition to the individual significance results on the datasets, we give a summary at the bottom of the table. For instance, in Table 2 the entry "0/12/14" in the column of MDL indicates that MDL significantly wins against RAND on 0 datasets and significantly loses on 14 datasets (on 12 datasets there is no significant difference). Similarly, the entry "4/0/22" in the column of MDL indicates the 'direct' wins/ties/losses (without significance tests).

The above results show that **overall RAND has the best performance** (in terms of RMSE and AUC) of all pruning criteria.

- RAND significantly outperforms another pruning criterion in 88 cases in total (i.e., 88 is the sum of all the significant losses in Tables 2 and 3). We explain the underlying causes for these cases in detail in the following sections.
- In contrast, RAND is significantly outperformed by some other criterion in only 6 cases (out of a total of 260). Three of these cases occur on the 'balance scale' dataset. For this dataset, the true concept is known but cannot be represented as a tree using our set of candidate tests. Hence the true concept needs to be approximated. The results show that, the larger the tree, the better the result. Hence, RAND is outperformed by EBP and NOPRUNING, which learn larger trees than RAND. For the three remaining cases in which RAND is significantly outperformed by some other criterion we do not have an explanation.

Because RAND performs so well, we use it as a reference to compare the other pruning criteria to. Having such a reference will make it easier to study the shortcomings of the other criteria.

Table 2 Root mean squared error (RMSE) for the six pruning criteria (lower is better).

	RAND	MDL	BIC	CHI	EBP	NOPRUNING
annealing	0.347	0.374 ○	0.370 ○	0.356	0.349	0.405 ○
australian credit	0.321	0.322	0.324	0.326	0.338	0.370 ○
balance scale	0.454	0.474 ○	0.463	0.464	0.437 ●	0.428 ●
breast wisconsin	0.201	0.212	0.195	0.201	0.203	0.204
chess kr-kp	0.081	0.105 ○	0.078	0.081	0.078	0.068
diabetes	0.419	0.422	0.422	0.420	0.435	0.460 ○
german credit	0.433	0.436	0.444	0.439	0.472 ○	0.488 ○
heart cleveland	0.401	0.415	0.407	0.402	0.414	0.427
ionosphere	0.281	0.276	0.280	0.291	0.277	0.284
mushroom	0.008	0.013	0.008	0.008	0.011	0.008
pen digits	0.272	0.336 ○	0.328 ○	0.286 ○	0.271	0.283 ○
primary tumor	0.853	0.894 ○	0.887 ○	0.848	0.873 ○	0.907 ○
segment	0.235	0.262 ○	0.253 ○	0.244 ○	0.238	0.249 ○
soybean	0.583	0.697 ○	0.684 ○	0.598 ○	0.593	0.629 ○
splice	0.224	0.223	0.222	0.228	0.235 ○	0.252 ○
thyroid	0.111	0.121 ○	0.113	0.112	0.111	0.139 ○
vehicle	0.464	0.502 ○	0.478	0.474	0.497 ○	0.522 ○
voting	0.196	0.196	0.195	0.195	0.192	0.199
vowel	0.630	0.740 ○	0.726 ○	0.663 ○	0.627	0.632
yeast	0.646	0.644	0.644	0.635 ●	0.665 ○	0.777 ○
biodegradability	0.451	0.460	0.444	0.453	0.451	0.451
cora	0.512	0.530	0.523	0.529 ○	0.530 ○	0.596 ○
diterpenes	0.694	0.721 ○	0.715 ○	0.684 ●	0.689	0.735 ○
gene	0.623	0.674 ○	0.659 ○	0.624	0.636 ○	0.656 ○
hiv	0.173	0.177 ○	0.174	0.175 ○	0.177 ○	0.186 ○
mutagenesis	0.442	0.471 ○	0.439	0.466	0.437	0.426
average	0.387	0.411	0.403	0.392	0.394	0.415
wins/ties/losses		4/0/22	8/1/17	4/1/21	9/0/17	4/1/21
significant w/t/l		0/12/14	0/18/8	2/18/6	1/17/8	1/9/16
average rank	2.17	4.69	3.29	3.13	3.12	4.60

4.4 Analysis of the Results

Below we analyze our experimental results in detail. Our aim is to gain insight into the differences in predictive performance between the various pruning criteria, and how these differences depend on the dataset characteristics. While we are eventually interested in the predictive performance (AUC and RMSE) of the learned trees, in the following sections we often start by analyzing the size of the trees. We do this because this allows us to explain differences in predictive performance between pruning criteria from a *causal* perspective: the pruning criterion directly determines the size of the tree, and the size of the tree in turn determines its predictive performance (symbolically: pruning criterion → tree size → RMSE/AUC).[6]

[6] Recall from Section 3.7 that the only difference between all the learning algorithms that we consider is the pruning criterion. Hence, if one algorithm performs worse than another one on

Table 3 Area under the ROC curve (AUC, in %) for the six pruning criteria (higher is better).

	RAND	MDL	BIC	CHI	EBP	NOPRUNING
annealing	89.0	83.1 ○	83.4 ○	85.8 ○	86.2	90.9
australian credit	91.5	91.6	91.6	91.2	91.5	89.9 ○
balance scale	87.4	84.9 ○	85.7 ○	85.7 ○	88.2	91.8 ●
breast wisconsin	97.7	96.5	98.0	97.8	97.1	98.1
chess kr-kp	99.9	99.7	99.9	99.9	99.8	99.9
diabetes	78.8	77.0	78.7	78.2	78.0	76.1
german credit	72.3	68.2 ○	72.0	71.5	71.4	71.0
heart cleveland	83.4	80.5	83.8	83.5	83.4	83.9
ionosphere	92.7	92.1	93.5	93.1	93.3	93.6
mushroom	100.0	100.0	100.0	100.0	100.0	100.0
pen digits	99.3	98.5 ○	98.6 ○	99.3	99.3	99.3
primary tumor	73.2	54.5 ○	59.6 ○	71.0	73.5	71.0
segment	99.5	98.7 ○	98.8 ○	99.4	99.4	99.5
soybean	96.3	90.1 ○	90.3 ○	95.4 ○	95.4	95.0 ○
splice	98.4	98.1	98.4	98.4	98.3	98.1 ○
thyroid	99.6	99.2	99.2	99.6	99.4	99.6
vehicle	89.9	86.5 ○	88.7	89.2	89.6	88.7
voting	98.2	97.5	98.3	98.3	97.6	98.3
vowel	92.9	86.0 ○	87.3 ○	91.9	92.2	92.1
yeast	80.1	78.2 ○	78.2 ○	80.0	79.1	75.2 ○
biodegradability	73.7	68.1	77.4	73.4	77.9	79.5 ●
cora	91.9	88.4 ○	89.1 ○	91.8	91.7	90.4 ○
diterpenes	85.6	69.5 ○	70.7 ○	82.0 ○	85.1	84.8
gene	86.8	77.1 ○	80.4 ○	86.8	86.0	85.3
hiv	75.4	68.6 ○	73.5 ○	75.1	63.6 ○	76.1
mutagenesis	74.6	73.8	75.3	74.4	73.1	78.6
average	88.8	84.9	86.6	88.2	88.1	88.7
wins/ties/losses		1/0/25	8/1/17	7/1/18	5/0/21	12/1/13
significant w/t/l		0/12/14	0/14/12	0/22/4	0/25/1	2/19/5
average rank	2.33	5.65	3.56	3.17	3.50	2.79

In each of the following sections, we consider one pruning criterion and analyze the situations in which this criterion fails while RAND behaves more robustly. This analysis will explain most of our experimental results, in particular the cases in which RAND outperforms some other pruning criterion (i.e., the many significant losses "○" in Table 2 and 3).

4.5 Analysis of the Performance of MDL and BIC

4.5.1 Results The performance of MDL and BIC is quite poor: they are significantly outperformed by RAND in respectively 28 and 20 cases, while the opposite

some dataset, this is really because it pruned too much or too little (in other words, because certain parts of the tree are too small or too large).

Table 4 Tree size for the six pruning criteria (lower is considered as better).

	RAND	MDL	BIC	CHI	EBP	NOPRUNING
annealing	18.2	4.2 ●	5.0 ●	9.8 ●	18.5	76.5 ○
australian credit	5.1	4.9	10.3 ○	9.9 ○	27.1 ○	89.0 ○
balance scale	21.2	10.6 ●	13.4 ●	13.4 ●	53.4 ○	128.9 ○
breast wisconsin	7.6	4.4 ●	10.1 ○	8.1	11.7 ○	26.0 ○
chess kr-kp	27.1	17.9 ●	27.8	26.4	27.8	41.4 ○
diabetes	10.9	3.9 ●	14.3	8.7	46.6 ○	179.6 ○
german credit	12.2	2.0 ●	29.8 ○	24.4 ○	103.9 ○	168.8 ○
heart cleveland	5.9	3.4 ●	14.0 ○	6.4	21.0 ○	48.5 ○
ionosphere	6.6	2.9 ●	14.5 ○	8.6	14.6 ○	20.1 ○
mushroom	9.8	9.3	9.8	9.8	9.5	9.8
pen digits	139.9	33.1 ●	37.0 ●	104.2 ●	213.1 ○	264.4 ○
primary tumor	14.5	0.5 ●	1.0 ●	5.7 ●	83.5 ○	140.3 ○
segment	35.1	11.1 ●	12.6 ●	27.6 ●	59.1 ○	86.1 ○
soybean	28.9	6.0 ●	6.1 ●	17.6 ●	68.7 ○	114.4 ○
splice	33.3	13.9 ●	23.4 ●	39.1 ○	79.6 ○	112.2 ○
thyroid	8.6	3.1 ●	4.2 ●	9.5	8.5	59.2 ○
vehicle	27.0	8.7 ●	13.8 ●	19.9 ●	100.2 ○	170.0 ○
voting	3.7	2.5	6.8 ○	5.1	6.8 ○	20.1 ○
vowel	75.2	8.6 ●	10.3 ●	34.1 ●	138.0 ○	153.9 ○
yeast	30.5	5.3 ●	5.3 ●	20.7 ●	119.2 ○	427.0 ○
biodegradability	5.6	1.9 ●	18.7 ○	7.9	31.3 ○	63.9 ○
cora	20.1	4.9 ●	5.8 ●	26.6 ○	94.2 ○	165.7 ○
diterpenes	42.0	2.1 ●	2.7 ●	12.4 ●	55.9 ○	111.9 ○
gene	30.5	6.4 ●	8.7 ●	40.8 ○	142.1 ○	210.8 ○
hiv	284.7	24.0 ●	63.0 ●	272.0	46.9 ●	2101.2 ○
mutagenesis	2.5	1.0	4.2 ○	2.0	8.3 ○	46.1 ○
average	34.9	7.6	14.3	29.6	61.1	193.7
wins/ties/losses		26/0/0	15/1/10	14/1/11	3/0/23	0/1/25
significant w/t/l		22/4/0	15/3/8	10/11/5	1/4/21	0/1/25
average rank	3.29	1.00	2.96	3.15	4.65	5.94

Table 5 Influence of the number of classes on the performance of MDL: percentage of datasets on which MDL is significantly outperformed by RAND (on all other datasets the differences between MDL and RAND are not significant).

Number of classes	AUC	RMSE
2	17%	25%
3-4	50%	75%
≥ 5	100%	80%

never happens. Tables 5 and 6 show that the results depend strongly on the number of classes in the dataset: the higher the number of classes, the poorer the performance of MDL and BIC. In other words, **MDL and BIC are less robust to a high number of classes than RAND**.

Table 6 Influence of the number of classes on the performance of BIC: percentage of datasets on which BIC is significantly outperformed by RAND (on all other datasets the differences between BIC and RAND are not significant).

Number of classes	AUC	RMSE
2	8%	0%
3-4	25%	0%
≥ 5	100%	80%

4.5.2 Explanation of the Results Also the results for tree size (Table 4) depend strongly on the number of classes: the higher the number of classes, the smaller the trees for MDL and BIC (as compared to RAND). This trend is explained by the complexity penalties used by MDL and BIC. Both complexity penalties grow linearly with the number of classes (Equation 2, p.10; Equation 4, p.10). When the number of classes increases, the penalties become so large that only very small trees are learned, resulting in poor predictive performance. No such effect plays for RAND because there is no explicit complexity penalty. This explains why RAND is more robust to a high number of classes than MDL and BIC.

4.5.3 Additional Experiments To support the above explanation, we performed additional experiments. We took seven datasets that each have at least 10 classes and manipulated these datasets by gradually reducing the number of classes (see Appendix A.3 for all details about the experimental setup). On six of the seven datasets the results confirm the above explanation (on the other dataset, 'yeast', there is no clear trend). The results on all datasets can be found in Appendix A.3, below we give the results on one dataset as a representative example.

Figure 4 shows the results on the 'diterpenes' dataset. In terms of **tree size** (Figure 4a) two trends are very clear.

- When increasing the number of classes, the trees for MDL and BIC become smaller and smaller, while the trees for RAND have roughly constant size. This is consistent with our above explanation: because the complexity penalties of MDL and BIC grow linearly with the number of classes, the penalties become so large that only very small trees are learned; no such effect plays for RAND because it has no explicit complexity penalty.
- Initially the tree for BIC is larger than for MDL, but this gap becomes smaller when increasing the number of classes. This can again be explained by the complexity penalties. The MDL-penalty is the same as the BIC-penalty (linear in the number of classes) plus an extra constant term. Hence, the tree for BIC is at least as large as for MDL. For a low number of classes, the gap between the two is substantial. When increasing the number of classes, this gap becomes smaller since the linear term in the MDL-penalty starts to dominate the constant term, making the MDL-penalty almost equal to the BIC-penalty.

Now consider the effect of tree size on **RMSE** and **AUC** (Figure 4b,c). For a low number of classes, the performance gap between RAND and MDL/BIC is relatively small (notice the overlapping error bars for RMSE). When increasing

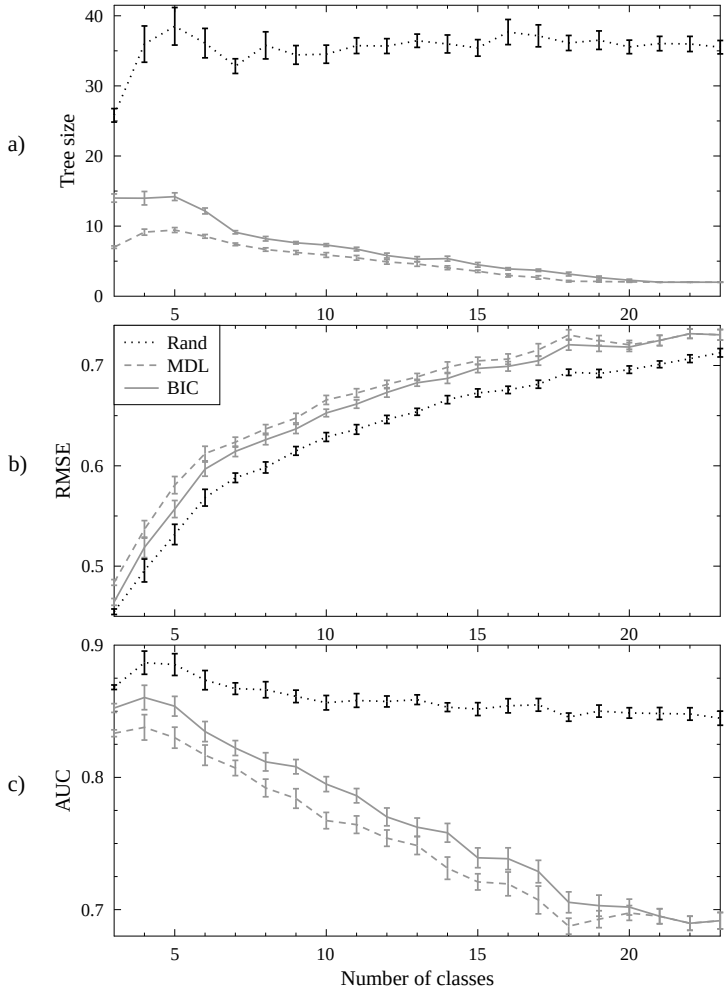

Fig. 4 Influence of the number of classes on tree size, RMSE and AUC, for the pruning criteria MDL, BIC and RAND. Error bars indicate 95% confidence intervals.

the number of classes, this gap grows rapidly: the trees for MDL and BIC become too small to perform well.

To summarize, the additional experiments confirm that MDL and BIC are less robust to a high number of classes than RAND due to the linear dependence of their complexity penalty on the number of classes.

4.5.4 Discussion The fact that MDL and BIC sometimes learn models that are too simple to perform well because of high complexity penalties is well-known (Quinlan & Rivest, 1989; Chickering & Heckerman, 1997; Domingos, 1998; Hastie

Table 7 Influence of the number of classes on the performance of NOPRUNING: percentage of datasets on which NOPRUNING is significantly outperformed by RAND.

Number of classes	AUC	RMSE
2	8%	33%
3-4	25%	75%
≥ 5	30%	90%

et al., 2001). However, to the best of our knowledge, we are the first to show the direct influence of the number of classes on this problem, and to show that randomization tests are much more robust to a high number of classes.

4.6 Analysis of the Performance of NOPRUNING

4.6.1 Results In terms of AUC, NOPRUNING is significantly outperformed by RAND on 5 datasets, while the opposite happens on 2 datasets. In terms of RMSE, NOPRUNING performs a lot worse: it is significantly outperformed by RAND on 16 datasets, while the opposite happens only once. Table 7 shows that the results again depend strongly on the number of classes in the dataset: the higher the number of classes, the poorer the RMSE of NOPRUNING . The same trend exists in terms of AUC, but much less pronounced.

Recall that RMSE evaluates the probability estimates themselves, whereas AUC evaluates only the probability-based rankings. Hence, the results essentially say that **NOPRUNING yields poor probability estimates** (but relatively good rankings), **especially when the number of classes is high**.

4.6.2 Explanation of the Results That NOPRUNING yields poor probability estimates can be explained by poor calibration. From all pruning criteria NOPRUNING is most susceptible to poor calibration. This is because NOPRUNING has the largest trees of all criteria, and hence the fewest examples in the leaves, and probabilities that are estimated from few examples are often poorly calibrated. As expected, this problem is more prominent when the number of classes is high, since then the number of examples per class can become very low in some leaves.

This explanation is also consistent with the above observation that the effect of the number of classes is much less pronounced in terms of AUC than in terms of RMSE. This is because AUC is less sensitive to the degree of calibration: differences in calibration that do not alter the resulting ranking do not affect AUC but do affect RMSE.

4.6.3 Additional Experiments To prove that the above explanation is indeed the true explanation for our results, we performed additional experiments in which we directly measured the calibration error. The calibration error is a measure of the difference between probability estimates and the corresponding fractions in the test data, see Appendix A.2 for the exact definition.

The experimental setup is exactly the same as in the previous section: we took seven datasets that each have at least 10 classes and manipulated these datasets by gradually reducing the number of classes. On five of the seven datasets the results confirm the above explanation (on the other datasets, 'gene' and 'soybean', there is no clear trend). The results on all datasets can be found in Appendix A.3, below we give the results on one dataset as a representative example.

Figure 5 shows the results on the 'diterpenes' dataset. When the number of classes increases, **tree size** (Figure 5a) stays roughly constant. Since the total number of examples was kept constant, this implies that the average number of examples of each class per leaf decreases. The effect on **calibration** (Figure 5b) is clear: the higher the number of classes, the poorer the calibration. This holds for both RAND and NOPRUNING. However, NOPRUNING runs into problems first because it has larger trees and hence a smaller number of examples of each class per leaf. Hence the gap in calibration error between RAND and NOPRUNING grows with the number of classes (Figure 5b).

The effect of the calibration problems on **RMSE** (Figure 5c) is very clear: when the number of classes increases, performance deteriorates for both RAND and NOPRUNING, but this goes faster for NOPRUNING. Hence the performance gap between RAND and NOPRUNING grows.

The effect of the calibration problems on **AUC** (Figure 5d) is similar but less pronounced than for RMSE. That the effect is less pronounced in terms of AUC was expected since AUC is less sensitive to poor calibration.

To summarize, the additional experiments show that the probability estimates of NOPRUNING are inferior to those of RAND due to poor calibration, especially when the number of classes is high.

4.6.4 Discussion Several techniques have been proposed to improve calibration of large probability trees (Zadrozny & Elkan, 2001; Wang & Zhang, 2006; Ferri, Flach, & Hernandez-Orallo, 2003b; Ling & Yan, 2003). We did not apply any of these recalibration techniques in this work.[7] The reason is that it is not clear which technique should be incorporated since it is unknown which of them performs best, and most of them require parameter tuning to perform well. Hence, choosing one of these recalibration techniques with a particular parameter setting and applying it in our experimental comparison is likely to yield results that are highly dependent on the exact choice made, limiting the generalizability of our conclusions. Investigating the influence of recalibration on the different pruning criteria is beyond the scope of this paper, and is more suited as the topic of a separate study.

4.7 Analysis of the Performance of EBP

4.7.1 Results In terms of AUC, EBP is significantly outperformed by RAND only once (the opposite never happens). In terms of RMSE, EBP performs worse: it is significantly outperformed by RAND on 8 datasets (the opposite happens once).

[7] Note however that the Laplace correction that we use (Section 2.2) can be seen as an extremely simple type of recalibration.

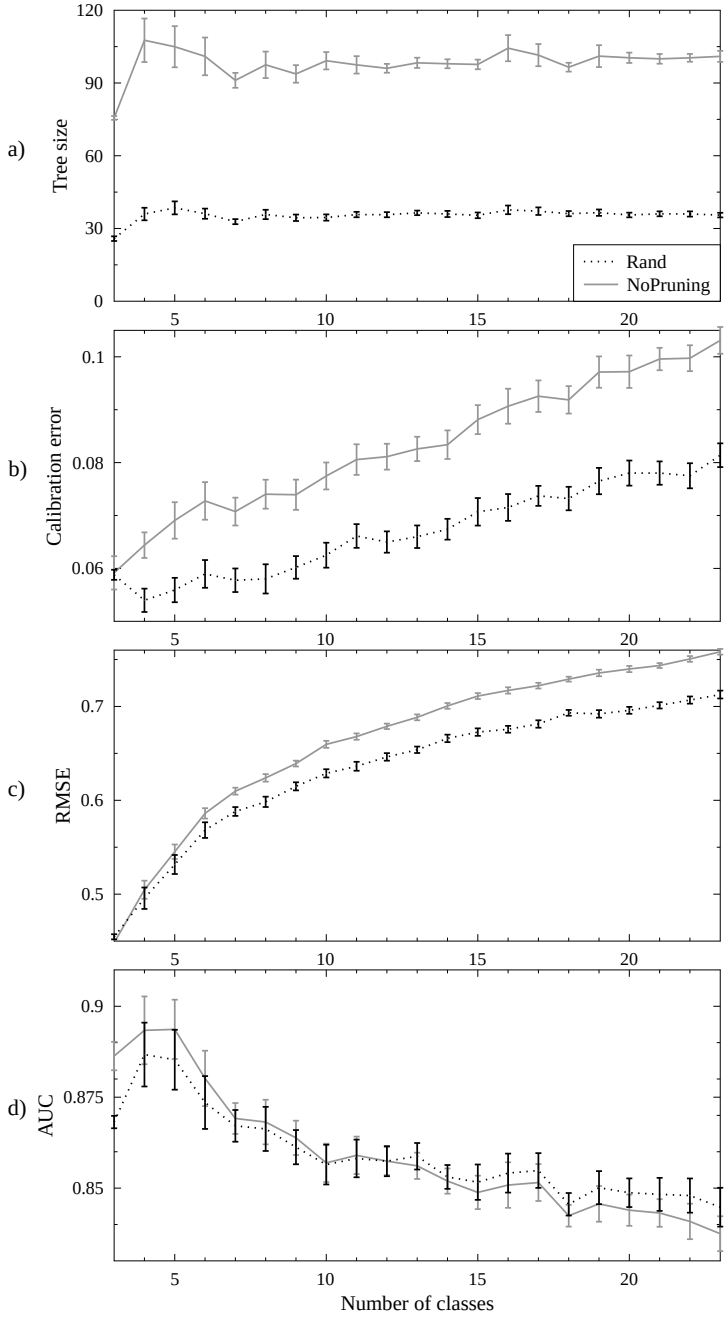

Fig. 5 Influence of the number of classes on tree size, calibration, RMSE and AUC for NOPRUNING and RAND. Error bars indicate 95% confidence intervals.

These results essentially say that usually **EBP yields poor probability estimates but good rankings**.

4.7.2 Explanation of the Results That EBP sometimes yields poor probability estimates but good rankings can be explained by poor calibration. From all pruning criteria other than NOPRUNING, EBP is the most susceptible to poor calibration. This is because EBP typically has the largest trees of all pruning criteria other than NOPRUNING (see Table 4, p.18) and hence the fewest examples in the leaves. On 7 of the 8 datasets on which EBP is significantly outperformed by RAND in terms of RMSE, the tree for EBP is indeed significantly larger than the tree for RAND, making EBP indeed more susceptible to poor calibration. The only dataset for which this explanation does not hold is the 'hiv' dataset which we discuss below.

4.7.3 Results (Part 2) There is only one dataset on which EBP is significantly outperformed by RAND in terms of both our performance measures (AUC and RMSE), namely the 'hiv' dataset. Interestingly, this is also the only dataset on which the tree for EBP is smaller than the tree for RAND, and it is by far the most strongly skewed dataset in our study. This suggests that **EBP is less robust to class skew than RAND.**

4.7.4 Explanation of the Results (Part 2) That EBP is not robust to class skew is well-known (Bradley, 1997; Zadrozny & Elkan, 2001). EBP tries to minimize the classification error, or in other words, to maximize the classification accuracy. It has been observed that on strongly skewed datasets EBP often learns a very small tree that almost always predicts the majority class (Bradley, 1997; Zadrozny & Elkan, 2001). While this might indeed give good classification accuracy, this usually gives poor probability estimates. This explains why EBP is not robust to class skew. Our results on the 'hiv' dataset suggest that RAND is more robust.

4.7.5 Additional Experiments To confirm that RAND is indeed more robust to class skew than EBP, we performed additional experiments. We took six two-class datasets and manipulated these datasets by gradually varying the class skew (see Appendix A.3 for all details about the experimental setup). On five of the six datasets the results followed the same trend (on the other dataset, 'breast wisconsin', there is no clear trend). The results on all datasets can be found in Appendix A.3, below we give the results on one dataset as a representative example.

Figure 6 shows the results on the 'hiv' dataset. It is clear that the class skew has a strong influence on the **tree size** (Figure 6a) for EBP. On nearly balanced datasets, EBP has larger trees than RAND. When increasing the class skew (going towards the left in the figure), trees for EBP become smaller at a much faster rate than happens for RAND. On strongly skewed datasets, trees for EBP are extremely small, even close to a single leaf.

The effect on **AUC** (Figure 6b) is very clear: on nearly balanced datasets EBP is competitive with RAND, but as the class skew increases EBP deteriorates and the performance gap with RAND grows.

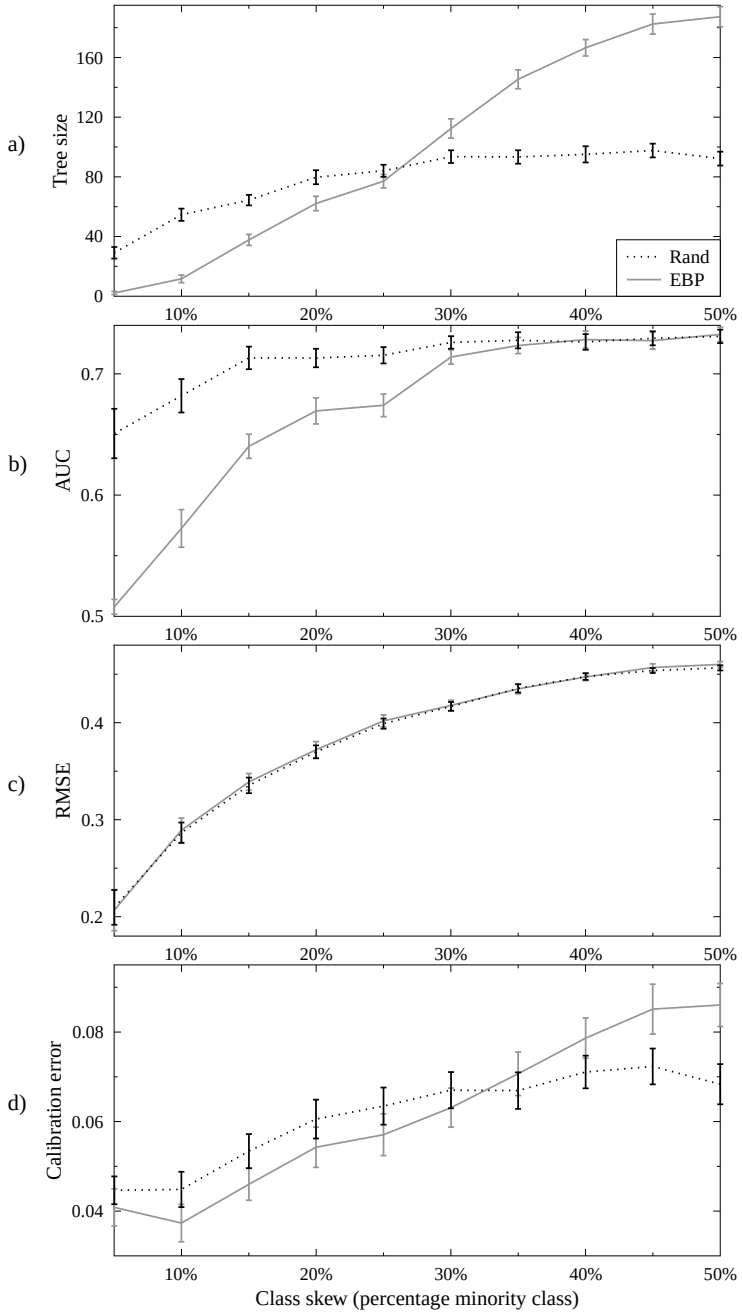

Fig. 6 Influence of the class skew on tree size, AUC, RMSE and calibration error, for the pruning criteria EBP and RAND. Error bars indicate 95% confidence intervals.

The effect on **RMSE** (Figure 6c) is much smaller: the performance gap between RAND and EBP is not strongly influenced by the class skew. This is because the decrease in tree size for EBP has two opposing effects on RMSE:

- a negative effect: as the tree for EBP becomes smaller, the probability estimates become less refined (hence also the poor AUC),
- a positive effect: as the tree for EBP becomes smaller, the calibration problem of EBP disappears and the probability estimates of EBP become better calibrated (see Figure 6d).

To summarize, the additional experiments confirm that EBP is less robust to class skew than RAND: the trees that are learned with EBP under strong class skew are too small to perform well in terms of AUC.

4.7.6 Discussion The fact that EBP has poor performance under strong class skew has been observed before on several real-world datasets (Bradley, 1997; Zadrozny & Elkan, 2001). The main contribution of our experiments is to verify this on manipulated data, and to show that RAND behaves more robustly.

4.8 Analysis of the Performance of CHI

4.8.1 Results Overall the CHI criterion performs rather well. Nevertheless there are still 10 cases in which it is significantly outperformed by RAND, while the opposite happens only in 2 cases.

4.8.2 Explanation of the Results The relationship between CHI and randomization tests has been studied in the literature before, both from a theoretical perspective and experimentally. Several disadvantages of CHI with respect to randomization tests have been identified. Hence, there are several possible explanations for the 10 cases in which RAND significantly outperforms CHI in our experiments.

First, RAND is more robust to small or skewed datasets than CHI (Frank & Witten, 1998). This is because the parametric sampling distribution used by CHI is not appropriate if there are not enough examples of each class in each node of the tree, which happens when the dataset is too small or too skewed. Such cases are no problem for RAND because RAND does not use a fixed parametric sampling distribution but instead estimates an empirical sampling distribution (which automatically takes into account any effects caused by a small or skewed dataset). Frank and Witten (1998) have shown experimentally that RAND is indeed more robust to small or skewed datasets than CHI.

Second, RAND is more robust to mutually correlated candidate tests than CHI (Jensen & Cohen, 2000). This is because the Bonferroni correction used by CHI is not correct when candidate tests are mutually correlated. Such cases are no problem for RAND because RAND does not need Bonferroni correction (the influence of the number of candidate tests is already taken into account in the empirical sampling distribution). Jensen and Cohen (2000) have shown experimentally that RAND is indeed more robust to mutually correlated tests than CHI.

Because of the multitude of possible causes (small dataset, strong class skew, correlated tests) we are not able to pinpoint the exact cause for each of the 10 cases in our experiments in which RAND significantly outperforms CHI.

4.9 Summary of Conclusions

We now briefly summarize our main conclusions. Overall RAND performs best: it outperforms the other pruning criteria much more often than that the opposite happens. To provide more insight into this result, we have identified and explained the shortcomings of the other pruning criteria by investigating several situations in which they fail (as compared to RAND).

- MDL and BIC are not robust to a high number of classes (because of the linear dependence of their complexity penalty on the number of classes).
- NOPRUNING yields probability estimates that are poorly calibrated, especially when the number of classes is high (because the trees are too large and hence do not have enough examples in each leaf).
- EBP is not robust to strong class skew, and also suffers from poorly calibrated probability estimates (but to a smaller extent than NOPRUNING).

Our experiments show that RAND is much less susceptible to these problems.

4.10 Additional Results

In addition to the experiments discussed in the previous sections, we performed several other experiments that are not at the heart of this paper but that further support our analysis. We now briefly summarize the results. For detailed results we refer to Appendix A.3.

First, in all our experiments we also measured performance in terms of **conditional log-likelihood (CLL)**, also known as *cross-entropy* (Friedman, Geiger, & Goldszmidt, 1997, Sect. 6.2). CLL is strongly correlated to RMSE but has the disadvantage that it is less robust to class noise (Caruana & Niculescu-Mizil, 2004). The reason why we nevertheless included CLL in our study is that is an often-used performance measure, for instance for Bayesian classifiers (Grossman & Domingos, 2004). We found that the results in terms of CLL are very similar to the results in terms of RMSE. All the conclusions that we had concerning RMSE also hold for CLL.

Second, in all our experiments we also measured performance in terms of **classification accuracy**. Since the focus of this paper is on probability estimation, classification accuracy is not of direct importance. Nevertheless, it is interesting to see whether the pruning criteria that are good for probability estimation are also good for classification. It turns out that the conclusions in terms of classification accuracy are slightly different than in terms of AUC and RMSE. The overall best criterion is no longer RAND but EBP. This is not surprising since EBP was designed to maximize classification accuracy and it is the most used criterion for

classification. RAND is the second best criterion and is significantly outperformed by EBP on 3 datasets.

Third, all the results reported in the previous sections are for experiments in which we used information gain as the **splitting criterion**, but we also repeated all experiments using gain ratio and Gini index instead. The results show that the choice of the splitting criterion does not influence our conclusions about the performance of the different pruning criteria. Moreover, our results show that the splitting criterion hardly influences the results at all. This is consistent with the conclusions of Ferri et al. (2003a, 2003b) who conducted extensive experiments to evaluate several splitting criteria (including some specifically tailored for probability estimation) and concluded that "differences between them are negligible".

Finally, the appendix also contains some more details for the experiments reported in this paper. First, whereas this paper only contains comparisons (wins/ties/losses) with RAND as a reference, the appendix also contains comparisons between all other possible pairs of pruning criteria. Second, whereas for the experiments on manipulated data (in which we vary the number of classes or the class skew) this paper contains only the results for the most relevant pruning criteria on one dataset, the appendix contains the results for all six pruning criteria on multiple datasets.

5 Related Work

In Section 1.1 we already mentioned the main previous work on pruning criteria for probability trees (Provost & Domingos, 2003; Ferri, Flach, & Hernandez-Orallo, 2003b; Neville et al., 2003; Friedman & Goldszmidt, 1998). In this section we discuss in more detail existing work about randomization tests for pruning decision trees (in the context of classification or probability estimation). There are two reasons why, from all pruning criteria in this paper, we discuss randomization tests in more detail. First, randomization is the best criterion in our experiments, and hence deserves most attention. Second, while randomization tests have been used before for pruning decision trees, most of these previous approaches have some subtle yet important differences with our approach.

We know of three approaches that are strongly related to our randomization approach, we discuss them below.

5.1 Randomization Tests in Relational Probability Trees

Neville et al. (2003) use randomization tests for prepruning relational probability trees. The main differences between their approach and our RAND criterion is that they use prepruning instead of postpruning, and that they randomize attribute values whereas we randomize class labels.

Randomizing attribute values instead of class labels yields the same trees on *attribute-value data*, but has the **advantage** that it corrects certain kinds of bias that often occur in *relational data* (for details, see Jensen et al. (2003)). Recall that our main conclusion from the experiments is that randomization performs best of all pruning criteria. If we would adapt our RAND criterion by also randomizing

attribute values instead of class labels, this could could further improve the results of randomization tests on the relational datasets, and hence could make our conclusion about the superiority of randomization tests even stronger.

A **disadvantage** of the approach of Neville et al. (2003) is that they use prepruning, which is inferior to postpruning (Fierens, Ramon, Blockeel, & Bruynooghe, 2007). A second disadvantage is that randomizing attribute values instead of class labels is computationally more expensive. Concretely, the resulting running time is $R + 1$ times the running time without randomization tests (R is the number of randomizations). In contrast, the running time of our RAND criterion is on average a lot smaller, as we will see in Section 6.3.2. This is the reason why we do not use their approach in our work.

5.2 Randomization Tests for Subtree Pruning

Oates and Jensen (1998) use randomization tests for subtree pruning. When a subtree starting in some internal node is considered for pruning, they randomize the class labels of all examples in that node, build a pruned tree on this randomized data, compute a score for this tree, and repeat this procedure R times. By storing all the computed scores for that node, an empirical sampling distribution for the score of a subtree under the null hypothesis is obtained. To decide whether the node should be pruned or not, the score of the actual subtree (build on non-randomized data) is compared to this empirical sampling distribution.

The **advantage** of their approach is that it might give slightly better results than our RAND criterion since they use subtree pruning whereas we use substump pruning. Hence, again it holds that applying this approach to our work could make our conclusion about the superiority of randomization tests even stronger.

The **disadvantage** of their approach is that it is computationally more expensive. Concretely, the running time of their approach is $R + 1$ times the running time without randomization tests, whereas the running time of our RAND criterion is on average a lot smaller (see Section 6.3.2). Again, this is the reason why we do not use their approach in our work.

5.3 Randomization Tests for Prepruning

Frank and Witten (1998) use randomization tests for prepruning. They use randomization to compute the correlation with the class variable for each candidate test individually, but they do not use Bonferroni correction to take into account the influence of the number of candidate tests.

Not using Bonferroni correction has the **disadvantage** of potentially overfitting when there are many candidate tests (the more tests, the more likely that at least one of them appears to be correlated with the class variable, even if in reality they are all uncorrelated). Note that in our CHI criterion we also compute the correlation for each candidate test individually but we do apply Bonferroni correction. In our RAND criterion we compute the correlation of the best candidate test (instead of the correlation for each candidate test individually), which makes

Bonferroni correction unnecessary and has the advantage of automatically taking into account the influence of not only the number of candidate tests but also of any potential mutual dependencies between candidate tests.

6 Computational Complexity of the Pruning Criteria

We now analyze the computational complexity of the pruning criteria used in this paper.

For each pruning criterion, the total time needed to learn a tree can be decomposed as follows

$$t_{total} = t_{unpruned} + t_{pruning}, \tag{6}$$

where $t_{unpruned}$ is the time for learning the unpruned tree, and $t_{pruning}$ is the time for postpruning this tree. Both quantities can be further decomposed into the time spent in each node of the tree.[8]

$$t_{unpruned} = \sum_{n \in nodes} t_{construct\_node}(n) \tag{7}$$

$$t_{pruning} = \sum_{n \in nodes} t_{prune\_node}(n) \tag{8}$$

Hence, the total running time can be decomposed accordingly.

$$t_{total} = \sum_{n \in nodes} \big(t_{construct\_node}(n) + t_{prune\_node}(n)\big) \tag{9}$$

The term $t_{prune\_node}(n)$ is negligible as compared to $t_{construct\_node}(n)$. This is because $t_{construct\_node}(n)$ depends on the number of examples and the number of candidate tests in node n (see below), while $t_{prune\_node}(n)$ does not: all information needed for deciding whether or not to prune a node is computed and stored during the construction step, hence accessing the dataset is not needed anymore during the pruning step. Hence, we get the following asymptotic approximation of the total running time.

$$t_{total} \simeq \sum_{n \in nodes} t_{construct\_node}(n) \tag{10}$$

The number of terms in the above summation is equal to the number of nodes in the unpruned tree. Unfortunately, it is hard to say in general how this number scales with the total number of examples and number of candidate tests (this is very dataset dependent). Fortunately, this issue is not relevant if we are only interested in *relative* differences in running time between the pruning criteria, since all pruning criteria that we consider use the same the unpruned tree. Hence, for the rest of our analysis, we do not consider the above sum, but only the individual terms $t_{construct\_node}(n)$. For ease of notation, we will leave the considered node n implicit and simply write $t_{construct\_node}$.

[8] If the pruning never reaches some node n (which can happen towards the root of the tree when doing substump pruning), then $t_{prune\_node}(n)$ is simply 0. Also, for NOPRUNING $t_{prune\_node}(n)$ is 0 for all nodes.

6.1 Pruning Criteria not Involving Randomization Tests

Let us first focus on the pruning criteria that do not involve randomization tests: CHI, MDL, BIC, EBP and NOPRUNING.

6.1.1 Decomposition of the Running Time For these criteria, $t_{construct\_node}$ can be decomposed as follows

$$t_{construct\_node} = |\mathcal{C}|\Big(N(t_{test} + t_{score}) + t_{heur}\Big) + t_{other}, \qquad (11)$$

where $|\mathcal{C}|$ is the number of candidate tests in the considered node, N is the number of examples in that node, t_{test} is the time for testing one example (i.e., for checking whether a candidate test succeeds on the example or not), t_{score} is the time for scoring one example (i.e., updating the class counts), t_{heur} is the time for computing the heuristic value (e.g. information gain) of a test given the class counts, and t_{other} is the remaining time, this is mainly the time for computing all the information needed to later do the pruning (computing the description length of the node, or the χ^2-score, etc.).

6.1.2 Discussion The first term in Equation 11 is the same irrespective of whether we use CHI, MDL, BIC, EBP or NOPRUNING. Only the second term, t_{other}, depends on exactly which of these pruning criteria we are using. However, this second term is independent of the number of examples and the number of candidate tests, and hence in practice this term is very small as compared to the first term. As a consequence, the differences in running time between CHI, MDL, BIC, EBP and NOPRUNING are very small. Indeed, in our experiments we found that the relative differences are never greater than 7%. The measured running times can be found in Appendix A.3.

6.2 The RAND *Criterion*

6.2.1 Decomposition of the Running Time The running time of RAND can be decomposed as follows:

$$t_{construct\_node} = |\mathcal{C}|\Big(N\big(t_{test} + (R+1)\, t_{score}\big) + (R+1)\, t_{heur}\Big) + t_{other}, \quad (12)$$

where R denotes the number of randomizations. In comparison with Equation 11, some terms in Equation 12 have been multiplied by a factor $R+1$. Concretely, t_{score} is multiplied by $R+1$. This is because in RAND we randomize the class labels of the examples, hence we need to score each example one time with the real class labels and R times with randomized class labels. Similarly, also t_{heur} is multiplied by $R+1$. In contrast, t_{test} is not multiplied by $R+1$. This is because randomizing the class label of an example does not change whether a given candidate test succeeds on the example or not, hence there is no need to re-check this for every randomization.

Table 8 Ratio of the running time of RAND to the running time of NOPRUNING as measured in our experiments. The average/minimum/maximum is over all respective datasets.

datasets	average	minimum	maximum
attribute-value	4.16	1.25	12.48
relational	1.75	1.23	2.56
all	3.60	1.23	12.48

6.2.2 Discussion As Equation 12 shows, the running time of RAND scales linearly in the number of randomizations R. However, this is of little practical importance since R is kept constant. In particular, larger datasets (more examples or more candidate tests) do not require a higher value of R. The only factor that influences R is the required significance level α for the randomization test (if one wants a lower α, this requires a higher R to keep estimates reliable). However, α is typically kept constant (Oates & Jensen, 1998; Jensen, Neville, & Rattigan, 2003; Neville et al., 2003; Jensen & Cohen, 2000; Frank & Witten, 1998), and hence also R is constant. In our experiments we always use $\alpha = 0.05$ and $R = 100$.

Let us now consider the computational overhead of RAND with respect to the other criteria, i.e., the ratio of the running time of RAND to that of the other criteria. Comparison of Equation 11 and Equation 12 shows that this overhead depends on the ratio of t_{test} to t_{score} (since t_{score} is multiplied by $R + 1$ while t_{test} is not). In practice, t_{test} is typically a lot larger than t_{score}. Hence, the overhead of RAND is in practice a lot smaller than a factor $R + 1$. This is confirmed by our experimental results, see Table 8 (this table shows a summary, full results are given in Appendix A.3). For instance, the average computational overhead of RAND with respect to NOPRUNING in our experiments is only a factor 3.6, which is indeed a lot smaller than a factor $R + 1 = 101$.[9]

6.3 Other Pruning Criteria based on Randomization Tests

In Section 5 we discussed two pruning criteria based on randomization tests by Neville et al. (2003) and Oates and Jensen (1998). As discussed, these two pruning criteria might have some advantage over our RAND criterion with respect to predictive performance. Below we show that this potential advantage comes at a high computational cost. Neither Neville et al. (2003) nor Oates and Jensen (1998) discuss the computational complexity or running time of their algorithm, so we are the first to analyze this.

6.3.1 Decomposition of the Running Time Let us first consider the pruning criterion of Neville et al. (2003) as discussed in Section 5.1. Although they use preprun-

[9] As Table 8 shows, the overhead of RAND is smaller on relational datasets than on attribute-value datasets. This is because candidate tests are usually more expensive (i.e., t_{test} is larger) for relational datasets than for attribute-value datasets, due to various issues in relational learning such as aggregation and lookahead (Van Assche et al., 2006; Blockeel & De Raedt, 1997).

ing, the running time of their algorithm can be decomposed in the same way as we did above for our postpruning criteria. The main difference between their criterion and our RAND criterion is that they randomize attribute values whereas we randomize class labels. The consequence of this is that for each candidate test and each example they need to check $R + 1$ times whether the test succeeds on the example while we only need to do this once (because randomizing the class label of an example does not change whether a given test succeeds on the example or not, but randomizing the attribute values of the example does). Hence, the time spent in a node by their algorithm can be decomposed as follows.

$$t_{construct\_node} = |\mathcal{C}|\Big(N\big((R+1)t_{test} + (R+1)\, t_{score}\big) + (R+1)\, t_{heur}\Big) + t_{other}, \tag{13}$$

$$= (R+1)|\mathcal{C}|\Big(N\big(t_{test} + \ t_{score}\big) + \ t_{heur}\Big) + t_{other} \tag{14}$$

Note that the main difference between Equation 13 and the equation for RAND (Equation 12) is that now t_{test} is also multiplied by $R+1$ while for RAND it isn't.

6.3.2 Discussion In Equations 11 to 14 t_{other} is negligible as compared to the other terms because it is independent of the number of examples N and the number of candidate tests $|\mathcal{C}|$ while the other terms aren't. Hence, Equation 14 implies that the running time of the algorithm of Neville et al. is $R + 1$ times the running time of NOPRUNING. In other words, the computational overhead of their algorithm with respect to NOPRUNING is a factor $R+1 = 101$. This is much higher than for RAND, which has an average overhead of a factor 3.6 in our experiments. Together this implies that the average expected overhead of their algorithm with respect to RAND on our datasets is a factor 28.1 ($= 101/3.6$). This high computational cost is the reason why we do not use the approach of Neville et al. in our work.

Let us now consider the pruning criterion of Oates and Jensen (1998) as discussed in Section 5.2. In this algorithm, when an internal node is considered for pruning, then for each randomization a new subtree needs to be build on the randomized data. Hence, the running time of this algorithm is $R+1$ times the running time of NOPRUNING. Again, this is much higher than for RAND, and this is the reason why we do not use the approach of Oates and Jensen in our work.

6.4 Summary

To summarize, the mutual differences in running time between all pruning criteria not involving randomization tests (CHI, MDL, BIC, EBP and NOPRUNING) are very small. RAND of course has a higher running time than these pruning criteria but the overhead is in practice relatively small (on average a factor 3.6 in our experiments) and is a lot smaller than for other pruning criteria based on randomization tests that have been proposed in related work.

To the best of our knowledge there is no existing work on randomization tests for pruning decision trees that discusses in detail the computational complexity or running times of the proposed algorithms. Hence, we are the first to show that randomization tests for decision tree pruning are computationally feasible in practice.

7 Conclusion

In this paper we gave a survey of six of the most important pruning criteria for probability trees, including a discussion of their theoretical advantages and disadvantages. We also performed an extensive experimental study of the relative performance of these pruning criteria.

Our experiments show that overall a pruning criterion based on randomization tests performs best: it outperforms the other pruning criteria much more often than that the opposite happens. To provide more insight into this result, we have identified and explained the shortcomings of the other pruning criteria by investigating several situations in which they fail. We found that the pruning criteria based on Minimum Description Length and Bayesian Information Criterion are not robust to a high number of classes (because of the linear dependence of their complexity penalty on the number of classes). Using unpruned trees has the disadvantage of giving probability estimates that are poorly calibrated (because the trees are too large), especially when the number of classes is high. Error-based pruning is not robust to strong class skew, and also suffers from poorly calibrated probability estimates (but to a smaller extent than for unpruned trees). Our experiments show that the pruning criterion based on randomization tests is much less susceptible to the above problems.

The drawback of the pruning criterion based on randomization tests is that it has a higher running time than the other criteria. Our recommendation is to use this criterion whenever its running time is not prohibitive. Otherwise, the best alternative depends on the characteristics of the dataset and the goal of the learning process. For instance, when the number of classes is low, the Bayesian Information Criterion typically performs quite well. When the goal is to obtain good rankings (rather than good probability estimates), learning unpruned trees is a good option too.

Acknowledgements Daan Fierens is supported by the Research Fund K.U.Leuven. This research was also supported by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT Vlaanderen), the Research Foundation Flanders (FWO Vlaanderen), GOA 2003/8 ‘Inductive Knowledge Bases’ and GOA/08/008 ‘Probabilistic Logic Learning’.

A Detailed Description of Experiments

A.1 Datasets

We use 20 attribute-value datasets and 6 relational datasets. All attribute-value datasets are from the UCI repository (Merz & Murphy, 1996). The relational datasets ‘biodegradability’, ‘diterpenes’, ‘mutagenesis’ and ‘hiv’ are standard datasets from the field of inductive logic programming (ILP). For more information on the first three datasets, see `http://www-ai.ijs.si/~ilpnet2/apps/`. The ‘hiv’ dataset is part of the NCI repository (Kramer et al., 2001). As is common for this dataset, we took the class labels “active” and “moderately active”

together since they are very infrequent. The two other relational datasets, 'cora' and 'gene', are standard datasets from the field of relational data mining (Neville et al., 2003). The 'cora' dataset is a subset of the papers collected by McCallum et al. (1999). We predict the high level topic of a paper and selected all papers for which at least one author, one reference and the number of pages are given, and that have a single topic. The 'gene' dataset is from the KDD Cup 2001, see `http://www.cs.wisc.edu/~dpage/kddcup2001/`. We predict the localization of a gene (the five least frequent localizations were discarded since together they only cover 13 examples).

To deal with missing values we preprocessed each dataset in the same way as Frank and Witten (1998): we first eliminated all attributes having more than 10% of missing values and then removed all examples for which the remaining attributes still had missing values.

A.2 Computation of the Calibration Error

For *two-class datasets*, we compute the calibration error in a similar way as Caruana and Niculescu-Mizil (2004) and Zadrozny and Elkan (2001). We sort all the examples in the test data according to their predicted probability of being positive. Then we divide the sorted examples into 10 bins each containing 10% of the examples (the first bin contains those examples that have the smallest predicted probability of being positive, etc.). For each bin i we compute two statistics:

- P_i, the average over all examples in bin i of the predicted probability of being positive,
- F_i, the fraction of examples in bin i that are truly positive.

For proper calibration, P_i should be close to F_i. Hence, the calibration error is defined as

$$\frac{\sum_{i=1}^{10} |P_i - F_i|}{10}.$$

For *multi-class datasets*, we compute a weighted average of the calibration errors obtained by using in turn each of the classes as positive and all others as negative, where the weight of a class is the fraction of examples of that class in the data. Note that this is the same strategy as for computing multi-class AUC (Section 4.1).

A.3 Full Experimental Results

The full results for all our experiments are available in an online appendix.
`http://www.cs.kuleuven.be/~dtai/pruning-ml09/`

References

Blockeel, H., & De Raedt, L. (1997). Lookahead and discretization in ILP. In *Proceedings of the 7th International Workshop on Inductive Logic Programming* (p. 77-85). Springer-Verlag.

Blockeel, H., & De Raedt, L. (1998). Top-down induction of first order logical decision trees. *Artificial Intelligence*, *101*(1-2), 285-297.

Bouckaert, R. (2004). Estimating replicability of classifier learning experiments. In *Proceedings of the 21st International Conference on Machine Learning.* Morgan Kaufmann.

Bradley, A. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, *30*, 1145–1159.

Caruana, R., & Niculescu-Mizil, A. (2004). Data mining in metric space: An empirical analysis of supervised learning performance criteria. In *Proceedings of the 10th International Conference on Knowledge Discovery and Data Mining* (p. 69-78). ACM Press.

Chickering, D., & Heckerman, D. (1997). Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning*, *29*(2-3), 181–212.

Cohen, P., & Jensen, D. (1997). Overfitting explained. In *Preliminary papers of the 6th International Workshop on Artificial Intelligence and Statistics* (p. 115-122).

Domingos, P. (1998). Occam's two razors: The sharp and the blunt. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining* (p. 37-43).

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In A. Prieditis & S. Russell (Eds.), *Proceedings of the 12th International Conference on Machine Learning* (p. 194-202). Morgan Kaufmann.

Džeroski, S., & Lavrač, N. (2001). *Relational Data Mining*. Springer-Verlag.

Esposito, F., Malerba, D., & Semeraro, G. (1997). A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(5), 476–491.

Fawcett, T. (2001). Using rule sets to maximize ROC performance. In *Proceedings of the 2001 IEEE International Conference on Data Mining* (p. 131-138). IEEE Press.

Ferri, C., Flach, P., & Hernandez-Orallo, J. (2003a). *Decision trees for ranking: Effect of new smoothing methods, new splitting criteria and simple pruning methods* (Tech. Rep.).

Ferri, C., Flach, P., & Hernandez-Orallo, J. (2003b). Improving the AUC of probabilistic estimation trees. In *Proceedings of 14th European Conference on Machine Learning.* Springer-Verlag.

Ferri, C., Hernández-Orallo, J., & Salido, M. A. (2003). Volume under the ROC surface for multi-class problems. In *Proceedings of 14th European Conference on Machine Learning* (p. 108-120). Springer-Verlag.

Fierens, D., Ramon, J., Blockeel, H., & Bruynooghe, M. (2005). A comparison of approaches for learning probability trees. In *Proceedings of 16th European Conference on Machine Learning* (p. 556-563).

Fierens, D., Ramon, J., Blockeel, H., & Bruynooghe, M. (2007). *A comparison of pruning criteria for probability trees* (Tech. Rep. No. CW 488). Department of Computer Science, Katholieke Universiteit Leuven.

Frank, E., & Witten, I. H. (1998). Using a permutation test for attribute selection in decision trees. In *Proceedings of the 15th International Conference on Machine Learning (ICML-98)* (pp. 152–160). Morgan Kaufmann.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, *29*, 131-163.

Friedman, N., & Goldszmidt, M. (1998). Learning Bayesian networks with local structure. In M. Jordan (Ed.), *Learning in Graphical Models* (pp. 421–459). Kluwer Academic Publishers.

Getoor, L., Friedman, N., Koller, D., & Pfeffer, A. (2001). Learning probabilistic relational models. In S. Džeroski & N. Lavrač (Eds.), *Relational data mining* (pp. 307–334). Springer-Verlag.

Grossman, D., & Domingos, P. (2004). Learning Bayesian network classifiers by maximizing conditional likelihood. In *Proceedings of 21st International Conference on Machine Learning*. Morgan Kaufmann.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The Elements of Statistical Learning*. Springer-Verlag.

Heckerman, D., Chickering, D., Meek, C., Rounthwaite, R., & Kadie, C. (2000). Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, *1*, 49-75.

Jensen, D., & Cohen, P. (2000). Multiple comparisons in induction algorithms. *Machine Learning*, *38*(3), 309–338.

Jensen, D., Neville, J., & Rattigan, M. (2003). *Randomization tests for relational learning* (Tech. Rep. No. 03-05). Department of Computer Science, University of Massachusetts.

Jensen, D., & Schmill, M. (1997). Adjusting for multiple comparisons in decision tree pruning. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining.* ACM Press.

Kersting, K., & De Raedt, L. (2007). Bayesian logic programming: Theory and tool. In *An Introduction to Statistical Relational Learning.* MIT Press.

Kramer, S., De Raedt, L., & Helma, C. (2001). Molecular feature mining in HIV data. In *Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining* (pp. 136–143). ACM Press.

Ling, C., & Yan, R. (2003). Decision tree with better ranking. In *Proceedings of the 20th International Conference on Machine Learning* (p. 480-487). Morgan Kaufmann.

McCallum, A., Nigam, K., Rennie, J., & Seymore, K. (1999). A machine learning approach to building domain-specific search engines. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence* (p. 662-667).

Mehta, M., Rissanen, J., & Agrawal, R. (1995). MDL-based decision tree pruning. In *Proceedings of the 1st International Conference on Knowledge Discovery*

and Data Mining (pp. 216–221).

Merz, C., & Murphy, P. (1996). *UCI repository of machine learning databases.* (http://www.ics.uci.edu/∼mlearn/MLRepository.html)

Neville, J., Jensen, D., Friedland, L., & Hay, M. (2003). Learning relational probability trees. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining.* ACM Press.

Oates, T., & Jensen, D. (1998). Large datasets lead to overly complex models: an explanation and a solution. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining.*

Provost, F., & Domingos, P. (2003). Tree induction for probability-based ranking. *Machine Learning*, *52*, 199–216.

Quinlan, J. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Quinlan, J., & Rivest, R. (1989). Inferring decision trees using the Minimum Description Length principle. *Information and Computation*, *80*, 227-248.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, *6*, 461–464.

Van Assche, A., Vens, C., Blockeel, H., & Džeroski, S. (2006). First order random forests: Learning relational classifiers with complex aggregates. *Machine Learning*, *64*(1-3), 149-182.

Wallace, C., & Patrick, J. (1993). Coding decision trees. *Machine Learning*, *11*, 7-22.

Wang, B., & Zhang, H. (2006). Improving the ranking performance of decision trees. In *Proceedings of the 17th European Conference on Machine Learning* (p. 461-472). Springer.

Zadrozny, B., & Elkan, C. (2001). Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In *Proceedings of the 18th International Conference on Machine Learning* (pp. 609–616). Morgan Kaufmann.