

A Comparison of Pruning Criteria for Probability Trees

Daan Fierens

Jan Ramon

Hendrik Blockeel

Maurice Bruynooghe

Report CW488, April 2007



Katholieke Universiteit Leuven

Department of Computer Science

Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

A Comparison of Pruning Criteria for Probability Trees

Daan Fierens

Jan Ramon

Hendrik Blockeel

Maurice Bruynooghe

Report CW488, April 2007

Department of Computer Science, K.U.Leuven

Abstract

Probability trees (or Probability Estimation Trees, PET's) are decision trees with probability distributions in the leaves. Usually decision trees are learned in a top-down manner with pre- or post-pruning. Which pruning criterion is used strongly influences the size of the resulting tree and the quality of the 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 or probability-based rankings. Hence, there is currently no clear view on the relative performance of all different pruning criteria for probability trees and it is unclear which criteria are preferable under which circumstances.

In this paper we survey six of the most important pruning criteria for probability trees. We discuss their theoretical advantages and disadvantages and we perform an extensive experimental study of their relative performance. The main conclusion is that a pruning criterion based on randomization tests usually performs best and learns trees that are relatively small. We identify several scenarios in which other pruning criteria achieve results comparable to those of randomization tests.

Keywords : Decision Trees, Pruning, Probability Estimation, Ranking, Randomization Tests

CR Subject Classification : I.2.6

1. Introduction

Probability trees (or Probability Estimation Trees, PET's) are decision trees with in the leaves probability distributions on a set of classes (Provost and 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 Bayesian networks (Friedman and Goldszmidt, 1998) and dependency networks (Heckerman et al., 2000). Second, they can be used for *ranking* instances according to the probability of belonging to a certain class (Provost and Domingos, 2003).

One of the key issues in learning probability trees, and decision trees in general, is how to determine the appropriate size of the tree. This is important for two reasons. First, the size of the tree directly influences the quality of the resulting probability estimates, since a tree that is too small underfits and a tree that is too large might overfit. Second, there are several situations in which the size of the tree is important in itself and where there is a preference for smaller trees. For example, sometimes the learned tree will be used by experts (such as medical doctors, see Ramon et al. (2006)) who want to be able to understand the tree, which is problematic if it is too large. Another example are algorithms that learn Bayesian networks or dependency networks by learning a set of probability trees (the attributes tested in the trees determine the edges in the network, see Heckerman et al. (2000); Ramon et al. (2007)). In this case, the size of the trees directly determines the complexity (number of edges) of the network and usually a simple network is desired. Hence, in this paper we consider small trees not only as a way to prevent overfitting but also as a goal in itself.

Usually decision trees are learned in a top-down manner and either prepruning or postpruning is applied. In prepruning a certain criterion is used during the top-down construction to decide whether it is still beneficial to continue growing the tree or whether a leaf should be created. In postpruning first a large tree is built by continuing to grow the tree as long as tests are available that partition the dataset. In a second phase the tree is then traversed in a bottom-up manner and subtrees that are not beneficial according to some criterion are removed.

For both prepruning and postpruning the choice of the exact pruning criterion strongly influences the size of the resulting tree and hence also the classification accuracy or the quality of the probability estimates. Moreover, it has been argued that to obtain good probability estimates or rankings one might have to use other pruning criteria than to obtain good classification accuracy (Provost and Domingos, 2003). While the effect of pruning criteria on classification accuracy has been well-studied for more than a decade (Esposito et al., 1997; Frank and Witten, 1998), only recently there is more interest in the effect on probability estimation or probability-based rankings (most work uses area under the ROC curve as an evaluation criterion). Provost and Domingos (2003) compare error-based postpruning with not using pruning. They conclude that neither of the two is significantly better than the other. Ferri et al. (2003b) introduce a new pruning method that prunes a leaf if the ratio of the number of examples in the leaf over the number of classes is below some threshold (the intuition is that probability estimates get less reliable if the number of examples is low or the number of classes is high). They conclude that despite its simplicity this method can reduce the size of the tree substantially without really degrading probability-based rankings. Neville et al. (2003) compare prepruning based

on randomization tests for relational data with prepruning based on chi-square tests. They conclude that randomization tests are preferable since they can adjust for some bias specific to relational learning. An older contribution is that of Friedman and Goldszmidt (1998) who define a pre- and postpruning criterion based on the Minimum Description Length principle and use it to learn Bayesian networks. None of the above contributions compare more than two pruning criteria at the same time. Hence there is no clear view on the relative performance of all different criteria. Concretely, it is currently unclear which pruning criteria for probability trees are preferable under which circumstances.

The goal of this paper is to gain insight into the relative performance of some of the most important (common or promising) pruning criteria for probability trees. The main contributions of this paper are twofold. First, we discuss six important postpruning criteria for probability trees and, when applicable, also their prepruning counterparts. This survey includes a discussion of theoretical advantages and disadvantages of the criteria. Second, we implemented these pruning criteria in a standard probability tree learner and perform an extensive experimental study of their relative performance. We identify and explain situations where certain criteria fail while other criteria behave more robustly. This leads to concrete recommendations about which criterion to use under which circumstances.

This paper is a continuation of previous work in which we experimentally compared a number of pruning criteria (Fierens et al., 2005b,c). Apart from providing more details about the criteria used and the results obtained, this paper significantly extends our previous work in three ways. First, this paper includes randomization tests as an extra 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 as the main evaluation criterion, we now in addition report conditional likelihood since this evaluates the probability estimates themselves rather than the probability-based rankings as area under the ROC curve does.

The remainder of this paper is structured as follows. In Section 2 we give some background on algorithms for pruning decision trees. In Section 3 we discuss how these algorithms can be used to learn probability trees. In Section 4 we discuss six of the most important pruning criteria for probability trees. In Section 5 we experimentally compare these pruning criteria. In Section 6 we discuss some related work. Finally, in Section 7 we conclude.

2. Decision Tree Pruning Algorithms

Decision trees (be it classification or probability trees) can be learned from a dataset of instances labelled with their true class. Decision trees are usually learned using the standard approach of top-down induction with either prepruning or postpruning. In this section we give some background on pruning algorithms which is needed for the rest of the paper. First we discuss how to construct an ‘unpruned’ tree, then we discuss prepruning and postpruning.

We want to be able to learn not only from attribute-value datasets such as the datasets in the UCI repository (Merz and Murphy, 1996), but also from relational datasets. For this reason we use Prolog queries as tests in the internal nodes of the trees. Since such tests are

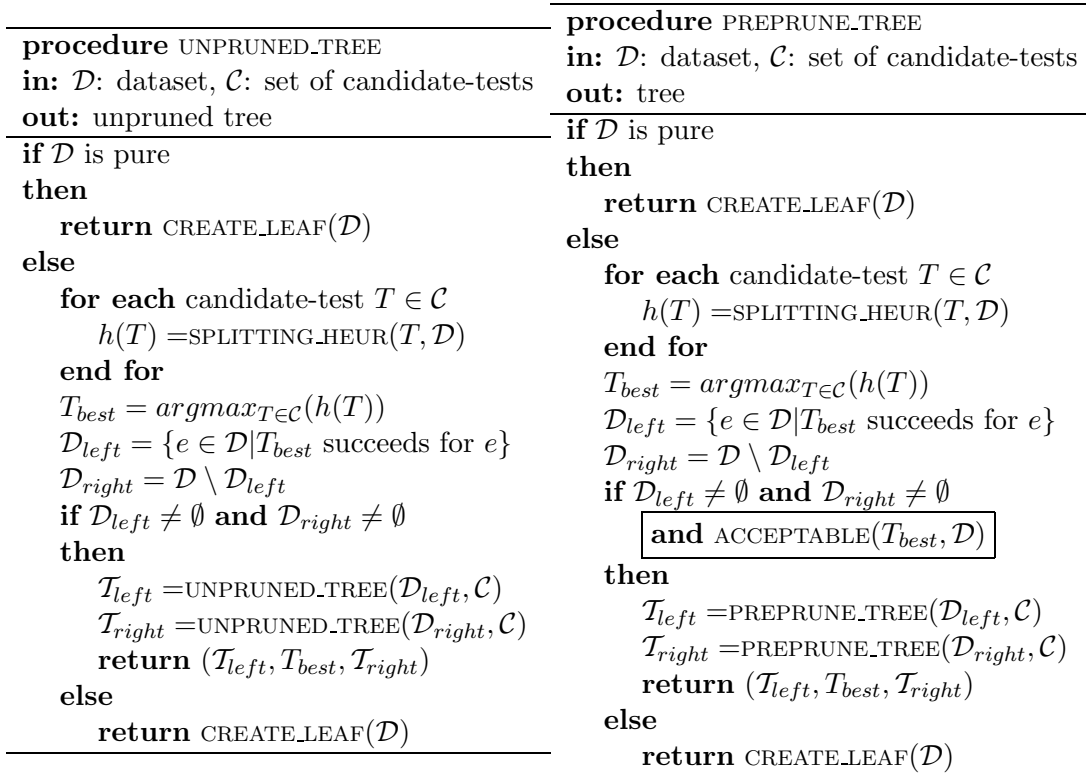


Figure 1: Algorithms for learning an unpruned decision tree (left) and for learning a decision tree using prepruning (right).

always binary (they either succeed or fail), we only consider binary trees in this paper. We further discuss the motivation and implications of this choice in Section 3.2.

2.1 Learning an Unpruned Tree

Learning an *unpruned tree* is usually done in a top-down manner. We are given a dataset and a set of candidate-tests and we start from the empty tree. For each candidate test we compute the heuristic value according to a splitting criterion. 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 (left).

The requirement that the best test effectively splits the dataset acts as a trivial stopping criterion. Note that the only cases in which the best test does not split the dataset are cases where none of the candidate tests split the dataset. This is because tests that do not split the dataset have a heuristic value of zero (for all common splitting criteria such as information gain, gain ratio, chi-square score, reduction in Gini index, ...) and we select the best test as the one with the highest heuristic value.

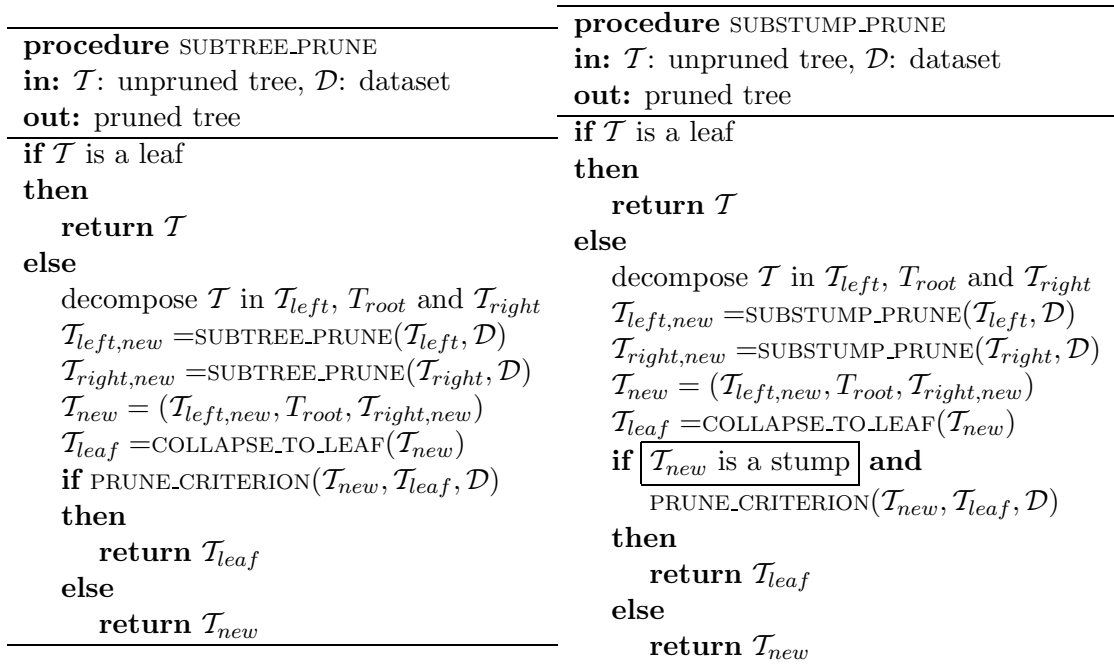


Figure 2: Algorithm for subtree postpruning (left) and substump postpruning (right).

2.2 Learning Decision Trees using Prepruning

Learning a tree using *prepruning* is very similar to learning an unpruned tree. The only difference is that now a stronger condition on T_{best} is used to decide to continue the tree building. We do not only require that T_{best} effectively splits the dataset but also that it is acceptable according to some prepruning criterion. We discuss prepruning criteria in detail in Section 4. The algorithm for prepruning is shown in Figure 1 (right), the difference with the algorithm for an unpruned tree is indicated by a box.

2.3 Learning Decision Trees using Postpruning

The advantage of prepruning is that it is usually quite fast. However, the disadvantage is that it sometimes gets stuck in a local optimum. To alleviate this problem one can apply postpruning instead of prepruning.

Learning a tree using *postpruning* is done in two phases. In the first phase we learn an unpruned tree, in the second phase we bottom-up postprune this tree. This postpruning can be done in two ways which we call subtree postpruning and substump postpruning (a substump is a subtree with exactly one internal node).

In *subtree postpruning* we first recursively prune the left and right subtree, resulting in a new tree \mathcal{T}_{new} . Then we use a postpruning 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} . The corresponding algorithm is shown in Figure 2 (left).

Substump postpruning is very similar to subtree postpruning. The only difference is that we only consider the possibility of replacing a tree \mathcal{T}_{new} by a leaf (depending on the pruning

criterion) if \mathcal{T}_{new} is a stump, but not if it is larger. The algorithm is shown in Figure 2 (right), the difference with subtree postpruning is indicated by a box.

Note that in each step of substump pruning we never remove more than a stump. However, since the pruning goes bottom-up, parts of a tree that were bigger than stumps initially can become stumps due to pruning and can subsequently be removed themselves. So substump pruning can remove multiple levels of a tree, but this requires multiple steps. In contrast, subtree pruning has the advantage that a subtree of arbitrary size can be removed in a single 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.

2.4 Relations between the Alternative Pruning Algorithms

There exist certain relations between the pruning algorithms discussed above that hold regardless of the exact pruning criterion used. Substump postpruning using a particular criterion never prunes more than subtree postpruning using that criterion, which in turn never prunes more than prepruning using that criterion. In other words, the tree for substump postpruning is at least as large as the tree for subtree postpruning, which is in turn at least as large as the tree for prepruning.

Substump postpruning never prunes more than subtree postpruning. To see this recall that subtree postpruning starts from the bottom of the tree and hence the first candidates for pruning are substumps. For these cases, subtree postpruning and substump postpruning are equivalent: if the one decides that a substump should be pruned (respectively retained), the other will decide the same. If it is decided that the substump should be pruned, both algorithms will continue pruning from the node higher up along that branch in the tree. However, if it is decided that the substump should be retained, subtree postpruning will still continue pruning from the node higher up along that branch in the tree, but substump postpruning will stop pruning along this branch. Hence, it is possible that the tree for substump postpruning is larger than the tree for subtree postpruning but not the other way around.

Subtree postpruning never prunes more than prepruning. Consider some tree \mathcal{T} and a ‘refined’ tree \mathcal{T}_{ref} that is obtained by replacing some leaf node N in \mathcal{T} by a stump. Suppose that \mathcal{T}_{ref} is better than \mathcal{T} according to the pruning criterion, then prepruning will not prune at node N (and return \mathcal{T}_{ref}). In such a case, subtree postpruning will also never prune at node N . This can be seen as follows. When node N is reached during the postpruning, \mathcal{T} (the tree that would be obtained if node N would be pruned) is compared to the tree resulting from postpruning below N . The latter is always a tree at least as good as \mathcal{T}_{ref} since postpruning only improves the quality of the trees (more specifically, this tree is either \mathcal{T}_{ref} itself or a tree larger and better than \mathcal{T}_{ref}). By transitivity such a tree is always better than \mathcal{T} and hence no pruning will occur at N .

3. Learning Probability Trees

The algorithms of the previous section are the generic algorithms for learning decision trees. They contain several parameters that have to be instantiated to obtain a concrete algorithm.

In this section we discuss how this is done for the case of probability trees. We also give some details about the tree learner we used in this paper.

3.1 Parameters of the Algorithms for Learning Decision Trees

The generic algorithms of Figure 1 and 2 contain three parameters that have to be instantiated to obtain a concrete algorithm for learning probability trees: a pre- or postpruning criterion, a splitting criterion (used in the `SPLITTING_HEUR` function) and a way to create leaf nodes (the `CREATE_LEAF` function). In this section we briefly discuss the latter two. We discuss pre- and postpruning criteria in detail in Section 4.

As a splitting criterion we use information gain. The reason is that some pruning criteria used in this paper are derived under this assumption, namely Minimum Description Length and Bayesian Information Criterion (see Section 4). In initial experiments we found that information gain performed slightly better than other common splitting criteria such as gain ratio and chi-square, but differences were very small. This is consistent with the conclusions of Ferri et al. (2003a,b) who conducted extensive experiments to evaluate several splitting criteria (including some specifically tailored for probability estimation) in terms of area under the ROC curve and concluded that “differences between them are negligible”.

For probability trees, creating a leaf means estimating the class probabilities in that leaf. It has been noticed repeatedly that it is crucial to use smoothing (Provost and Domingos, 2003; Ferri et al., 2003b). Without smoothing, 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 and Domingos, 2003; Ferri et al., 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 et al., 2003b), curtailment (Zadrozny and Elkan, 2001), shrinkage (Wang and Zhang, 2006) and others (Ling and Yan, 2003), but they typically require tuning parameters.

3.2 Learning Probability Trees using Tilde

In our work we are not only interested in learning from attribute-value data where each instance is described by a fixed vector of attributes such as the datasets in the UCI repository (Merz and Murphy, 1996), but also from relational datasets (Džeroski and Lavrač, 2001) where instances have a more complex, relational structure such as molecules (Kramer et al., 2001) or citation networks (McCallum et al., 1999). In relational learning, probabilistic approaches such as upgrades of Bayesian networks (Getoor et al., 2001; Kersting and De Raedt, 2007; Fierens et al., 2005a) currently attract a lot of attention. As a consequence, probability estimation is becoming 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 data, we implemented the algorithms for learning probability trees in the relational decision tree learner Tilde (Blockeel and De Raedt, 1998; Van Assche et al., 2006). Tilde uses as a representation for an instance a

first-order logic interpretation, basically 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. Tilde with its standard settings was developed as a relational version of the well-known C4.5 tree learner (Quinlan, 1993). Hence, when applied on attribute-value data Tilde coincides with C4.5 except for one difference: Tilde uses only binary tests whereas C4.5 can also use n -ary tests ($n \geq 2$). If we have a ternary attribute that can have the value a , b or c , this will give rise to three binary tests in Tilde (testing whether it is a or not, b or not and c or not).

4. Probability Tree Pruning Criteria

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. One criterion, the Bayesian Information Criterion, is an exception since to our knowledge it has not been applied to probability trees before. Nevertheless, we include this criterion since its general principles are well-known (Domingos, 1998) and since it is strongly related to (but usually better than) Minimum Description Length, another criterion we consider. Together the criteria included in this paper offer a good coverage of the different families of pruning criteria. First, from the criteria using statistical tests for pairwise correlations we include chi-square tests (Section 4.1) and randomization tests (Section 4.2). Second, from the information-theoretic or complexity-based criteria we include Minimum Description Length (Section 4.3) and Bayesian Information Criterion (Section 4.4), but there are several others (for example Minimum Message Length, see Wallace and Patrick (1993)). Third, from the criteria based on classification accuracy we include Error Based Pruning (Section 4.5), but there are many others (Esposito et al., 1997). Finally we also consider the option of not pruning at all (Section 4.6) since this has been suggested as a good approach for probability trees (Provost and Domingos, 2003).

For each criterion we discuss its background, how it can be applied in postpruning (and when applicable prepruning) and what its advantages and disadvantages are.

4.1 Statistical Tests for Pairwise Correlation (Chi-square test)

4.1.1 BACKGROUND

Suppose that we have a set of candidate tests and we want to check whether the best test T_{best} is really correlated with the class variable. This can be done using any standard hypothesis test for pairwise correlation such as the χ^2 (chi-square) -test. Concretely, we formulate the null hypothesis that T_{best} is not correlated with the class variable and check whether we can reject this null hypothesis. We reject the null hypothesis, and hence accept the test T_{best} , if the χ^2 -score of T_{best} with respect to the class variable is significant at the level $\alpha = \frac{0.05}{N_{test}}$ with N_{test} the number of candidate tests. Adjusting the α -level to the number of candidate tests N_{test} is known as the Bonferroni correction. This is necessary to avoid the ‘multiple tests problem’ or ‘multiple comparisons problem’ (Jensen and Cohen,

2000; Cohen and Jensen, 1997): the score of the best test T_{best} from a set of candidate tests generally increases as the number of candidate tests increases, even if they are all uncorrelated with the class variable; hence not taking into account the number of candidate tests increases the probability of wrongly accepting T_{best} .

4.1.2 THE PRUNING CRITERIA

The above approach can be used for **prepruning** in a straightforward way. Such a prepruning strategy has been proposed before by Neville et al. (2003). This prepruning strategy can easily be turned into a **substump postpruning** strategy as follows. In the first phase we learn an unpruned tree but while doing so we also compute in each step the acceptability of T_{best} as above and store this result for later use (we do not use it to decide about stopping the tree building). In the second phase we then use substump postpruning where we replace a stump by a leaf if the test in that stump was unacceptable and retain the stump otherwise. This substump postpruning strategy has been proposed before by Jensen and Schmill (1997). In the remainder we refer to this criterion (with prepruning or substump postpruning) as CHI.

4.1.3 DISCUSSION

This CHI criterion has some disadvantages that arise from the assumptions typically made in the application of hypothesis tests. There are three main such assumptions. First, the Bonferroni correction for computing the α -level assumes that all candidate tests are mutually independent. When this is not the case, the Bonferroni correction ‘overcorrects’ the α -level, thereby increasing the probability of wrongly rejecting the best test (Jensen and Cohen, 2000). As a consequence, trees might be learned that are too small. Second, to determine the required χ^2 -score for a given α -level, the sampling distribution of the χ^2 -score needs to be used. In theory, however, 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 and Witten, 1998). Third, it is assumed that examples are independent, which might be violated in relational datasets. Since this last assumption is made by all pruning criteria in this paper, we do not discuss it any further.

4.2 Randomization Tests

4.2.1 BACKGROUND

Randomization is a general technique to avoid assumptions typically made in the application of hypothesis tests such as the chi-square test. To decide whether the best test from a set of candidate tests is really correlated with the class variable, we essentially need the sampling distribution for the heuristic value of the best test under the null hypothesis that none of the candidate tests are correlated with the class variable. The idea behind randomization is to construct this sampling distribution empirically (Jensen and Cohen, 2000; Cohen and Jensen, 1997). To do so we randomly permute the class labels of all examples (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 $h(T_{best}^{rand})$. By repeating this procedure

N_{rand} times we obtain an empirical distribution for $h(T_{best})$ under the null hypothesis. This distribution can then be used to decide about the acceptability of the actual best test T_{best} (computed with non-permuted class labels). We use $N_{rand} = 100$. With an α -level of 0.05, we accept T_{best} if $h(T_{best})$ is greater than at least 95% of the $h(T_{best}^{rand})$ values we stored. Note that here the Bonferroni correction is not needed because the influence of the number of candidate tests is already included in the empirical sampling distribution.

4.2.2 THE PRUNING CRITERIA

Randomization tests can be used for **prepruning** in the same way as the CHI criterion from the previous section. This has been proposed before by Frank and Witten (1998) and Neville et al. (2003) (although they use a somewhat different type of randomization, see Section 6). Randomization tests can also be used for **substump postpruning**, again in the same way as for the CHI criterion. As far as we know, this has not been used before. In the remainder we refer to this criterion (with prepruning or substump postpruning) as **RAND**.

4.2.3 DISCUSSION

The advantage of **RAND** is that it is very flexible: it avoids the first two assumptions made by the CHI criterion. As such 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 expensive since for each candidate test a heuristic value needs to be computed N_{rand} times instead of a single time like for CHI. However, this does not imply that **RAND** is a factor N_{rand} slower than CHI. The reason is the following. For CHI we basically have to perform two computations for each candidate test: first we compute the result of the candidate test on each example (i.e. whether the test succeeds or not on the example), second we use these results together with the class labels of the examples to compute the heuristic value (information gain) of the candidate test. For **RAND** we have to do the second computation N_{rand} times but the first computation only once. The reason for this is that we randomize class labels and the first computation is independent of the class labels but the second computation is not. According to our experiments, this second computation is quite fast as compared to the first computation. Hence, the computational overhead for **RAND** with respect to CHI is typically a lot smaller than a factor N_{rand} , see Section 5.3.6.

4.3 Minimum Description Length (MDL)

4.3.1 BACKGROUND

Minimum Description Length (MDL) is a general criterion for determining the quality of a model. MDL aims to find a balance between the fit of the model to the training data and the complexity of the model. The MDL criterion for probability trees developed by Friedman and Goldszmidt (1998) is as follows. The score of a probability tree on a dataset is the sum of the description length of the tree and the description length of the data given the tree: $score(\mathcal{T}, \mathcal{D}) = DL(\mathcal{T}) + DL(\mathcal{D} | \mathcal{T})$ (lower scores are better). 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 $DL(\mathcal{T})$ can be seen as a complexity penalty for the tree \mathcal{T} . It is defined as $1 + 0.5(C - 1)\log_2(N)$ if \mathcal{T} is a leaf and as $1 + \log_2(N_{test}) + DL(\mathcal{T}_{left}) + DL(\mathcal{T}_{right})$ otherwise, with \mathcal{T}_{left} and \mathcal{T}_{right} the subtrees of \mathcal{T} . The description length $DL(\mathcal{D} | \mathcal{T})$ of the data \mathcal{D} given a probability tree \mathcal{T} is defined as $\sum_{i=1}^N -\log_2(p_{\mathcal{T}}(c_i|e_i))$, where $p_{\mathcal{T}}(c_i|e_i)$ is the probability according to \mathcal{T} that the example e_i has its true class label c_i . Using arguments similar to those of Friedman and Goldszmidt (1998), this sum can be rewritten as $\sum_l N_l H(\mathcal{D}_l)$, where the sum ranges over all leaves of the tree and N_l denotes the number of examples in leaf l and $H(\mathcal{D}_l)$ denotes the entropy of these examples with respect to the class variable.¹

4.3.2 THE PRUNING CRITERIA

This MDL criterion can be used for **prepruning**. In prepruning we have to decide whether the current tree \mathcal{T} is better than a refined tree \mathcal{T}_{ref} obtained by replacing a leaf node l in \mathcal{T} with a stump containing the best test T_{best} for that node. To decide this we could in principle simply compare the score of both trees. If $score(\mathcal{T}_{ref}, \mathcal{D}) < score(\mathcal{T}, \mathcal{D})$ we add the test and continue the tree building, otherwise we create a leaf. However, since $score(\mathcal{T}, \mathcal{D})$ and $score(\mathcal{T}_{ref}, \mathcal{D})$ typically have a lot of terms in common that cancel each other, this can be done more efficiently (Friedman and Goldszmidt, 1998). More precisely, we only have to check the condition $N_l IG(T_{best}, \mathcal{D}_l) > 2 + \log_2(N_{test}) + 0.5(C - 1)\log_2(N)$ where $IG(T_{best}, \mathcal{D}_l)$ is the information gain of T_{best} on the examples in l . This is an intuitive result: we only add a test if the increase in fit of the data is higher than the increase in complexity due to adding the test (which has the net effect of adding one internal node and one leaf node).

This prepruning strategy could of course be turned into a substump postpruning strategy in the same way as we did for the CHI and RAND criteria. Unlike these two criteria, however, MDL can also easily be used for **subtree postpruning**. To decide whether a given tree is better than the same tree with some subtree replaced by a leaf, we can simply compute the scores of both trees and select the tree with the lowest score. Again we do not have to compute the full scores since they have a lot of terms in common that cancel each other. Both the above prepruning strategy and this subtree postpruning strategy have been proposed by Friedman and Goldszmidt (1998). We refer to this paper for more details on the exact formulation of the criteria.

In the postpruning strategy we apply an optimization that was not included in the work of Friedman and Goldszmidt (1998). Concretely, we speed up the first step, the construction of an unpruned tree. The optimization is based on the fact that sometimes it can be derived from the properties of the MDL criterion that a subtree will definitely be pruned in the second step, even before actually building this subtree in the first step. In such cases we immediately create a leaf in the ‘unpruned’ tree instead of first building this subtree. We discuss the exact condition for such cases in Appendix A.

1. Other MDL criteria have been proposed for classification trees (Quinlan and Rivest, 1989; Mehta et al., 1995) but these are not applicable to probability trees since they consider the case where a leaf contains a single class label instead of a distribution on the class labels, which leads to different complexity penalties.

4.3.3 DISCUSSION

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

4.4 Bayesian Information Criterion (BIC)

4.4.1 BACKGROUND

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 and Goldszmidt, 1998; Hastie et al., 2001, sect. 7.8). Since we already discussed MDL, we focus on the latter interpretation. 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, but in BIC it only depends on the number of independent numerical parameters. To compute the description length of a probability tree BIC only takes into account the probabilities in the leaves, whereas MDL would also take into account the tests in the internal nodes. As a consequence, the complexity penalties for BIC are smaller than those for MDL.

While the BIC criterion in itself is well-known (Domingos, 1998) and is widely applied to certain kinds of models (such as Bayesian networks, see Chickering and Heckerman (1997)), it appears we are the first to apply it to probability trees. The BIC criterion we propose for probability trees is as follows. We define $DL(\mathcal{T})$ as $0.5(C - 1)\log_2(N)$ if \mathcal{T} is a leaf (note that $C - 1$ is the number of independent parameters in a leaf) and as $DL(\mathcal{T}_{left}) + DL(\mathcal{T}_{right})$ otherwise, where \mathcal{T}_{left} and \mathcal{T}_{right} are the subtrees of \mathcal{T} . The definition of the description length $DL(\mathcal{D} | \mathcal{T})$ of the data given the model is the same as for MDL.

4.4.2 THE PRUNING CRITERIA

This BIC criterion can be used for **prepruning** in the same way as the MDL criterion in the previous section. Precisely, we add a test T_{best} to a node l in a tree if $N_l IG(T_{best}, \mathcal{D}_l) > 0.5(C - 1)\log_2(N)$, and create a leaf otherwise. This prepruning strategy can be turned into a **subtree postpruning** strategy, again in the same way as for the MDL criterion (and with a similar optimization, see Appendix A).

4.4.3 DISCUSSION

As already mentioned, the complexity penalties for BIC are smaller than those for MDL. This can usually be seen as an advantage of BIC since MDL tends to overpenalize models and the smaller complexity penalties often cause BIC to perform better than MDL. Nevertheless, also BIC often selects models that are too simple because complexity penalties are still too high (Chickering and Heckerman, 1997; Hastie et al., 2001, sect. 7.7).

4.5 Error Based Pruning (EBP)

4.5.1 BACKGROUND

Error Based Pruning (EBP) tries to minimize the estimated number of errors a tree would make on unseen data (Quinlan, 1993). To estimate this number, the training data in a leaf 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 for l . Precisely, let N_l denote the total number of examples in leaf l and E_l denote the actual number of errors on the training data in a leaf l (this is the number of training examples in l that do not have the most frequent class in l). The error rate e_l on the training data in l is defined as the fraction $\frac{E_l}{N_l}$. Assuming 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, sect. 4.3). Call u_l the upper bound of this confidence interval, this is then the estimated error rate on unseen data. The estimated number of errors of l is then the product $u_l N_l$.

4.5.2 THE PRUNING CRITERION

The above idea can be used for **subtree postpruning** 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. The estimated number of errors of a subtree is computed as the sum of the estimated number of errors of all its leaves. This is one of the most widely used postpruning strategies (Esposito et al., 1997). It is also the pruning strategy used by the C4.5 system (Quinlan, 1993).² However, there is one difference between EBP as explained above and EBP as implemented by the C4.5 system: 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 postpruning 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.

4.5.3 DISCUSSION

EBP was developed for classification trees and tries to minimize the classification error. A disadvantage of EBP is that it does not work well on datasets with a very skewed class distribution: in such cases EBP typically learns a very small tree that almost always predicts the majority class (Bradley, 1997; Zadrozny and Elkan, 2001; Fierens et al., 2005c). While this can indeed result in good classification accuracy, this usually gives poor probability-based rankings. Hence, for probability trees EBP is expected not to be the best choice. Nevertheless, we include EBP in our comparison because it is so widely used for classification trees but also for probability trees (Provost and Domingos, 2003; Ferri et al., 2003b).

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

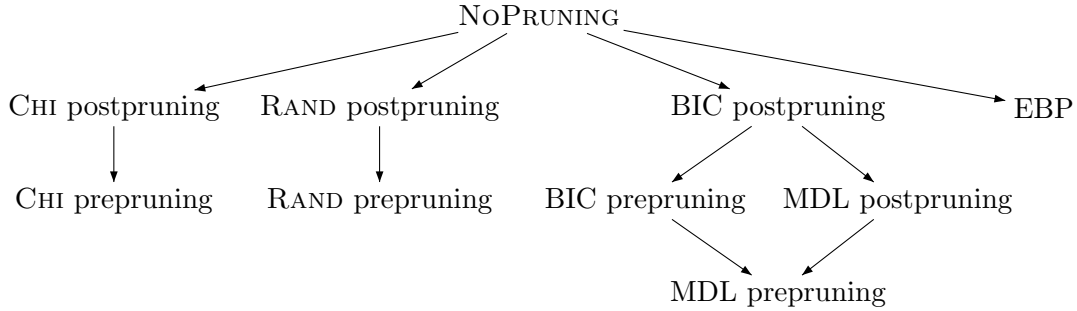


Figure 3: Relations between the size of the trees learned using the different pruning criteria. A directed arc from a criterion C_1 to a criterion C_2 denotes that the tree for C_1 is at least as large as the tree for C_2 .

4.6 No Pruning

4.6.1 THE PRUNING CRITERION

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

4.6.2 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. They also notice, however, that there is the competing effect that such large trees typically have few examples in the leaves which can result in overfitting (high variance in the estimated probabilities and hence poor performance on unseen examples). Another disadvantage of unpruned trees is that they are not really interpretable since they are so large.

4.7 Relations Between the Six Criteria

To further clarify the above pruning criteria we now discuss some relations that hold between the trees learned by these criteria on the same dataset.

First and most obvious, all criteria (with both pre- and postpruning) learn trees at most as large as the tree for NOPRUNING. Second, BIC never prunes more than MDL since the complexity penalty for BIC is strictly smaller than that for MDL. In other words, the tree for BIC prepruning (respectively postpruning) is at least as large as the tree for MDL prepruning (respectively postpruning). Third, as explained in Section 2, trees for postpruning with a particular criterion are always at least as large trees for prepruning with that criterion. This holds for all criteria for which we consider both prepruning and postpruning: CHI, RAND, MDL and BIC.

All the above relations are summarized graphically in Figure 3.

5. Experiments

In this section we experimentally compare the pruning criteria from the previous section. In Section 5.1 we discuss the datasets and evaluation criteria used and the experimental setup. In Section 5.2 we experimentally compare prepruning with postpruning for the CHI, RAND, MDL and BIC criteria. In Section 5.3 we experimentally compare the six postpruning criteria.

5.1 Datasets, Evaluation Criteria and Experimental Setup

First we discuss the datasets used, then the evaluation criteria used and finally the experimental setup.

5.1.1 DATASETS

We use 20 attribute-value datasets and 6 relational datasets. Table 1 gives the most important characteristics of these datasets.

All attribute-value datasets are from the UCI repository (Merz and Murphy, 1996). 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.

The relational datasets biodegradability, diterpenes and mutagenesis are standard ILP datasets.³ The cora dataset is a subset of the database of 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 2001.⁴ We predict the localization of a gene (the five least frequent localizations were discarded since together they only cover 13 examples). The hiv dataset is part of the NCI repository (Kramer et al., 2001). As is common for this dataset, the class labels ‘active’ and ‘moderately active’ were taken together since they are very infrequent.

For a discrete attribute we include a candidate test for each possible value (for example, a ternary attribute leads to three tests, see Section 3.2). For a continuous attribute, we use minimal entropy discretization (Dougherty et al., 1995) to generate seven thresholds, and then use candidate tests of the form ‘attribute value is smaller than threshold’.

5.1.2 EVALUATION CRITERIA

We use five evaluation criteria: area under the ROC curve, conditional likelihood, classification accuracy, tree size and running time.

Area under the ROC curve (AUC) is used to measure the quality of probability-based rankings (Provost and Domingos, 2003). For two-class problems AUC is the probability that a random positive example and a random negative example are ranked correctly (the positive example is predicted as more likely to be positive than the negative example). For multi-class problems, we use a weighted average of the AUC’s obtained by using in turn

3. Information on these datasets is available on <http://www-ai.ijs.si/~ilpnet2/apps/>.

4. Information on this dataset is available on <http://www.cs.wisc.edu/~dpage/kddcup2001/>.

	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

Table 1: Characteristics of the datasets: number of examples, number of classes, skew (fraction of examples of 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.

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 et al., 2003c; Fawcett, 2001).

Conditional Log-Likelihood (CLL) of a probability tree \mathcal{T} on a dataset is the logarithm of the probability of the class labels in the dataset according to the tree: $\log_2(\prod_{i=1}^N p_{\mathcal{T}}(c_i|e_i)) = \sum_{i=1}^N \log_2(p_{\mathcal{T}}(c_i|e_i))$, with $p_{\mathcal{T}}(c_i|e_i)$ the probability according to \mathcal{T} that the example e_i has its true class label c_i . In this paper we always report the *normalised CLL*, which is the CLL as above divided by the number of examples N . It can be shown that maximizing the CLL of a model on a dataset is the same as minimizing the KL-divergence between the conditional probability distribution on the class labels specified by the data and the conditional distribution specified by the model (Friedman et al., 1997, Sect. 6.2).

Most existing work on probability trees uses AUC as an evaluation criterion. To the best of our knowledge, we are the first to use CLL. This is somewhat surprising since CLL is a common evaluation criterion for Bayesian classifiers (Grossman and Domingos, 2004), which basically serve the same purpose as probability trees. The reason why we use both AUC and CLL is that they are complementary. Whereas AUC measures the quality of probability-based rankings, CLL measures the quality of the probability estimates themselves. AUC only depends on the relative ordering of the estimated probabilities but not on whether the probabilities are calibrated (whether they correspond to fractions in the data, see Zadrozny and Elkan (2001)). Hence, when probability estimates are poorly calibrated, AUC can still be high (Provost and Domingos, 2003) but CLL will always be low.

For completeness we also briefly report results for *classification accuracy*. This allows us to see whether the pruning criteria that are good for probability estimation are also good for classification.

We also use *tree size* as an evaluation criterion. As already motivated in Section 1, we consider small trees not only as a way to prevent overfitting, but also as a goal in itself. We measure tree size as the number of internal nodes of the tree (this is the number of leaves minus one since we always learn binary trees).

Finally we also report *running time*: the time in seconds to build a tree measured on a Pentium 4 1.70GHz machine with 1GB RAM.

5.1.3 EXPERIMENTAL SETUP

For running time we report the time to build a tree on an entire dataset. For all other evaluation criteria, we report results that are averages of ten different runs of five-fold cross-validation on a dataset.

When comparing two pruning criteria, we report ‘direct’ wins, ties and losses as well as significant wins, ties and losses. To compute direct wins, ties and losses, we do not use any statistical test: a criterion wins if its average result (over the runs of the cross-validation) is better. To compute *significant* wins, ties and losses, we use two-tailed paired t-tests: a criterion wins if it is significantly better according to the t-test (with $\alpha = 0.05$). To compute the t-tests we use the ‘averaging over sorted runs’ approach of Bouckaert and Frank (2004) since this avoids the usual problems with applying t-tests on repeated experiments, such as an elevated type I error and low replicability. When comparing several pruning criteria we also report *average ranks* over all datasets (the best criterion on a dataset has rank 1 on that dataset, the second best rank 2, ...)

5.2 Experimental Comparison of Prepruning versus Postpruning

We experimentally compared prepruning with postpruning for the CHI, RAND, MDL and BIC criteria. Numerical results for all evaluation criteria are given in an online appendix (Fierens et al., 2007). We now state the most important conclusions.

As explained in Section 4.7, for each of the considered criteria the tree for postpruning is at least as large as the tree for prepruning. For the MDL and BIC criteria, we observed in our experiments that the tree for postpruning is only significantly larger for a few datasets (1 and 3 datasets out of 26, respectively), on the other datasets there is no significant difference. For the CHI and RAND criteria, it happens more often that the trees for postpruning are significantly larger than the trees for prepruning (12 and 16 datasets, respectively).

We also investigated whether the additional tree size for postpruning results in better probability estimates or not. The conclusions were uniform across the four criteria and are as follows. For AUC postpruning is usually slightly better than prepruning. For CLL postpruning is sometimes slightly better, sometimes slightly worse. However, for both AUC and CLL the differences are almost always small and are only significant for one dataset, namely hiv.

On the hiv dataset, for each pruning criterion postpruning is significantly better than prepruning (for both AUC and CLL). Moreover, the choice of postpruning versus prepruning has a bigger impact than the choice of the pruning criterion: for instance, the worst AUC for postpruning is still better than the best AUC for prepruning. This suggests that on hiv prepruning gets stuck in a local optimum. Since hiv is by far the largest dataset in our experiments we investigated whether the better performance of postpruning is due to the size of the dataset. To do so we constructed learning curves for all pre- and postpruning criteria, but we found no clear evidence that the difference in performance between postpruning and prepruning depends on the size of the dataset.

The main drawback of postpruning is that it is computationally more expensive than prepruning. For MDL and BIC running times of postpruning are on average a factor 2.5 higher than running times of prepruning, for CHI a factor 3.6 and for RAND a factor 2.0.

Overall we conclude that, although postpruning is computationally more expensive, it is preferable over prepruning since it is generally more ‘reliable’ than prepruning (it is never significantly worse but sometimes significantly better). This is consistent with the common knowledge about pruning for classification trees.

5.3 Experimental Comparison of Postpruning Criteria

In this section we experimentally compare the different pruning criteria. Since we concluded from the previous section that postpruning is preferable over prepruning, we now focus on postpruning criteria. Concretely, we mutually compare the six postpruning criteria from Section 4. First we discuss tree size and the quality of the probability estimates (AUC and CLL). Next, we briefly discuss some additional experiments to clarify some of the earlier observations. Then we briefly discuss classification accuracy and running times. Finally, we give some overall conclusions.

5.3.1 TREE SIZE

We start by analysing the size of the trees learned. Table 2 gives the detailed results (since the number of classes turns out to be important for our conclusions, we report it for each dataset between brackets and we sort the datasets accordingly). In this table we use RAND as a reference against which we compute wins, ties and losses since RAND is overall the best criterion, as we will see below. The symbol “•” next to a result for a 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). For tree size better means smaller. In addition to the individual significance results on the datasets, we give a summary at the bottom of the table. For instance, the entry “22/4/0” in the column of MDL indicates that MDL significantly wins against RAND on 22 datasets and significantly loses on 0 datasets (on 4 datasets there is no significant difference). In a similar way, the entry “26/0/0” in the column of MDL indicates the ‘direct’ wins, ties and losses (without significance tests, see Section 5.1.3).

The main observations are the following.

1. Trees for MDL are by far the smallest. This holds for each of the 26 datasets (hence the average rank of 1.0).
2. Trees for BIC (average rank 2.96) are larger than for MDL when the number of classes is low and of similar size otherwise.
3. On average, trees for CHI (average rank 3.15) and RAND (average rank 3.29) are somewhat larger than for BIC. However, when the number of classes is low, trees for BIC are larger.
4. Trees for EBP (average rank 4.65) are usually larger than for MDL, BIC, CHI and RAND, except on skewed datasets.
5. Trees for NOPRUNING are of course always the largest. The average difference with the other criteria goes from a factor 3.2 (for EBP) to a factor 25.5 (for MDL).

We now briefly discuss and explain some of these observations.

As mentioned in Section 4.3 the complexity penalties for MDL are usually so high that very simple models are learned. This seems to be confirmed by our results. In Section 4.4 we noted that an advantage of BIC is that complexity penalties are smaller than for MDL and hence less simple models are learned. Indeed, when the number of classes is low, the trees for BIC are considerable larger than for MDL. However, when the number of classes increases, differences become smaller. This is explained as follows. The complexity penalty for BIC is linear in the number of classes, whereas the complexity penalty for MDL is the same plus some additional terms independent of the number of classes. Hence, with an increasing number of classes, the term linear in the number of classes will start to dominate the other terms in the MDL penalty and the MDL penalty will get closer to the BIC penalty. We investigate the influence of the number of classes in more detail in Section 5.3.4.

Trees for CHI and RAND are larger than for BIC if the number of classes is high. However, when the number of classes is low, trees for BIC are larger. This is explained by the fact

	RAND	MDL	BIC	CHI	EBP	NOPRUNING
australian credit (2)	5.1	4.9	10.3 ◦	9.9 ◦	27.1 ◦	89.0 ◦
biodegradability (2)	5.6	1.9 ●	18.7 ◦	7.9	31.3 ◦	63.9 ◦
breast wisconsin (2)	7.6	4.4 ●	10.1 ◦	8.1	11.7 ◦	26.0 ◦
chess kr-kp (2)	27.1	17.9 ●	27.8	26.4	27.8	41.4 ◦
diabetes (2)	10.9	3.9 ●	14.3	8.7	46.6 ◦	179.6 ◦
german credit (2)	12.2	2.0 ●	29.8 ◦	24.4 ◦	103.9 ◦	168.8 ◦
heart cleveland (2)	5.9	3.4 ●	14.0 ◦	6.4	21.0 ◦	48.5 ◦
hiv (2)	284.7	24.0 ●	63.0 ●	272.0	46.9 ●	2101.2 ◦
ionosphere (2)	6.6	2.9 ●	14.5 ◦	8.6	14.6 ◦	20.1 ◦
mushroom (2)	9.8	9.3	9.8	9.8	9.5	9.8
mutagenesis (2)	2.5	1.0	4.2 ◦	2.0	8.3 ◦	46.1 ◦
voting (2)	3.7	2.5	6.8 ◦	5.1	6.8 ◦	20.1 ◦
balance scale (3)	21.2	10.6 ●	13.4 ●	13.4 ●	53.4 ◦	128.9 ◦
splice (3)	33.3	13.9 ●	23.4 ●	39.1 ◦	79.6 ◦	112.2 ◦
thyroid (4)	8.6	3.1 ●	4.2 ●	9.5	8.5	59.2 ◦
vehicle (4)	27.0	8.7 ●	13.8 ●	19.9 ●	100.2 ◦	170.0 ◦
annealing (5)	18.2	4.2 ●	5.0 ●	9.8 ●	18.5	76.5 ◦
segment (7)	35.1	11.1 ●	12.6 ●	27.6 ●	59.1 ◦	86.1 ◦
cora (10)	20.1	4.9 ●	5.8 ●	26.6 ◦	94.2 ◦	165.7 ◦
gene (10)	30.5	6.4 ●	8.7 ●	40.8 ◦	142.1 ◦	210.8 ◦
pen digits (10)	139.9	33.1 ●	37.0 ●	104.2 ●	213.1 ◦	264.4 ◦
yeast (10)	30.5	5.3 ●	5.3 ●	20.7 ●	119.2 ◦	427.0 ◦
vowel (11)	75.2	8.6 ●	10.3 ●	34.1 ●	138.0 ◦	153.9 ◦
soybean (15)	28.9	6.0 ●	6.1 ●	17.6 ●	68.7 ◦	114.4 ◦
primary tumor (21)	14.5	0.5 ●	1.0 ●	5.7 ●	83.5 ◦	140.3 ◦
diterpenes (23)	42.0	2.1 ●	2.7 ●	12.4 ●	55.9 ◦	111.9 ◦
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 2: Tree size (number of internal nodes) for the six postpruning criteria.

that the complexity penalty for BIC is linear in the number of classes (resulting in small trees if the number of classes is high) while no such effect plays for CHI or RAND.

Trees for EBP are almost always larger than for MDL, BIC, CHI and RAND. The only significant exceptions are on the hiv dataset where the tree for EBP is considerable smaller than for BIC, CHI and RAND. This atypical behaviour of EBP on hiv is not surprising since hiv is a strongly skewed dataset and it is known that EBP learns extremely small trees on such datasets (Bradley, 1997; Zadrozny and Elkan, 2001). We performed an additional experiment in which we generated samples of the hiv dataset with varying skewness, see the online appendix (Fierens et al., 2007). The goal of this experiment was to verify the hypothesis that the atypical behaviour of EBP on the original hiv dataset is indeed due to its skewness. We found that on less skewed samples of the hiv dataset, trees for EBP are larger than for MDL, BIC, CHI and RAND. This indeed confirms our hypothesis.

5.3.2 QUALITY OF THE PROBABILITY ESTIMATES: AUC

We now discuss the quality of the probability estimates in terms of AUC. We will relate the results for AUC (and CLL in the next section) to the results for tree size since tree size is the only factor influencing the probability estimates for which the various pruning criteria differ. Results for AUC are shown in Table 3.

The main observations are the following.

1. Overall RAND (average rank 2.33) is the best criterion to maximize AUC. The only criterion that sometimes significantly outperforms RAND is NOPRUNING (average rank 2.79). This happens on 2 datasets. On the other hand, RAND significantly outperforms NOPRUNING on 5 datasets.
2. MDL and BIC have poor performance on datasets with a high number of classes. However, on most datasets with a low number of classes BIC has performance comparable to RAND. The latter also holds for MDL but to a smaller extent (on fewer datasets).
3. EBP is significantly outperformed by RAND on only one dataset namely hiv, a very skewed dataset.

NOPRUNING is the only criterion that sometimes significantly outperforms RAND, namely on the balance scale and biodegradability datasets. For balance scale this can be explained by the fact that the true concept (which is known) cannot be represented as a tree using our set of candidate tests. Hence the concept needs to be approximated and this appears to be better possible for larger trees (indeed, for balance scale sorting the criteria for tree size is the same as sorting them for AUC). For biodegradability it is possible that a similar explanation holds but we have no concrete evidence for this.

BIC is significantly outperformed by RAND on 12 datasets. On the 14 other datasets there is no significant difference (on 8 of them BIC is slightly better, on 6 RAND is slightly better). Interestingly, these 14 datasets almost all have a low number of classes whereas the other 12 datasets almost all have a high number of classes. This coincides with our observations for tree size: if the number of classes is high, trees for BIC are significantly smaller than for RAND (due to the complexity penalty for BIC), while otherwise they are

PRUNING CRITERIA FOR PROBABILITY TREES

	RAND	MDL	BIC	CHI	EBP	NoPRUNING
australian credit (2)	91.5	91.6	91.6	91.2	91.5	89.9 ◦
biodegradability (2)	73.7	68.1	77.4	73.4	77.9	79.5 ●
breast wisconsin (2)	97.7	96.5	98.0	97.8	97.1	98.1
chess kr-kp (2)	99.9	99.7	99.9	99.9	99.8	99.9
diabetes (2)	78.8	77.0	78.7	78.2	78.0	76.1
german credit (2)	72.3	68.2 ◦	72.0	71.5	71.4	71.0
heart cleveland (2)	83.4	80.5	83.8	83.5	83.4	83.9
hiv (2)	75.4	68.6 ◦	73.5 ◦	75.1	63.6 ◦	76.1
ionosphere (2)	92.7	92.1	93.5	93.1	93.3	93.6
mushroom (2)	100.0	100.0	100.0	100.0	100.0	100.0
mutagenesis (2)	74.6	73.8	75.3	74.4	73.1	78.6
voting (2)	98.2	97.5	98.3	98.3	97.6	98.3
balance scale (3)	87.4	84.9 ◦	85.7 ◦	85.7 ◦	88.2	91.8 ●
splice (3)	98.4	98.1	98.4	98.4	98.3	98.1 ◦
thyroid (4)	99.6	99.2	99.2	99.6	99.4	99.6
vehicle (4)	89.9	86.5 ◦	88.7	89.2	89.6	88.7
annealing (5)	89.0	83.1 ◦	83.4 ◦	85.8 ◦	86.2	90.9
segment (7)	99.5	98.7 ◦	98.8 ◦	99.4	99.4	99.5
cora (10)	91.9	88.4 ◦	89.1 ◦	91.8	91.7	90.4 ◦
gene (10)	86.8	77.1 ◦	80.4 ◦	86.8	86.0	85.3
pen digits (10)	99.3	98.5 ◦	98.6 ◦	99.3	99.3	99.3
yeast (10)	80.1	78.2 ◦	78.2 ◦	80.0	79.1	75.2 ◦
vowel (11)	92.9	86.0 ◦	87.3 ◦	91.9	92.2	92.1
soybean (15)	96.3	90.1 ◦	90.3 ◦	95.4 ◦	95.4	95.0 ◦
primary tumor (21)	73.2	54.5 ◦	59.6 ◦	71.0	73.5	71.0
diterpenes (23)	85.6	69.5 ◦	70.7 ◦	82.0 ◦	85.1	84.8
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

Table 3: Area Under ROC Curve (%) for the six postpruning criteria.

bigger. This suggests that the linear dependency of the complexity penalty for BIC on the number of classes is probably too strong and hence on datasets with a high number of classes the learned trees are too small to perform well.

For MDL the dependency of performance on the number of classes also holds but to a somewhat smaller extent than for BIC (MDL is outperformed by RAND on two more datasets than BIC and these datasets have a low number of classes, namely 2 and 4). When comparing the performance of MDL to the other criteria, we see that MDL often has the worst performance of all criteria, hence its average rank of 5.65. Recall from the previous section that MDL has the smallest trees of all criteria due to the high complexity penalty used. This suggests that the complexity penalty for MDL is effectively too high to give good performance.

EBP is significantly outperformed by RAND only on one dataset, namely hiv. Moreover, on this dataset EBP has by far the worst AUC of all criteria. Again this coincides with our observations about tree size: hiv is the only dataset on which the tree for EBP is smaller than the tree for RAND, which was expected because of the strong skewness of hiv. On the other 25 datasets the trees for EBP are larger than for RAND but this additional tree size apparently does not lead to a better AUC since EBP never significantly outperforms RAND.

CHI is significantly outperformed by RAND on 4 datasets. On each of these 4 datasets the tree for RAND is bigger than the tree for CHI (while in total this is only the case on half of the datasets). This suggests that on these 4 datasets CHI prunes too much.

A common pattern in the above observations is that criteria that build trees smaller than the tree for RAND on a dataset often have a significantly worse AUC than RAND on that dataset. Usually these are cases where the criterion has some deficit on that dataset (such as BIC on datasets with a high number of classes or EBP on strongly skewed datasets). On the other hand, criteria that build trees approximately as large or larger than the tree for RAND rarely have a significantly better AUC than RAND. This suggests that RAND finds a good balance between underfitting and overfitting.

5.3.3 QUALITY OF THE PROBABILITY ESTIMATES: CLL

We now discuss CLL. Results for CLL are shown in Table 4. For convenience we show negative normalised CLL (lower is better).

The main observations are the following.

1. Overall RAND (average rank 1.87) is the best criterion to optimize CLL. There is only one criterion (namely NOPRUNING) that sometimes significantly outperforms RAND but this happens only on one dataset. That RAND is the best criterion is in line with the conclusions for AUC but for CLL this is even more pronounced (better rank and less losses for RAND).
2. CHI (average rank 2.63) is the second best criterion to optimize CLL. CHI is significantly outperformed by RAND on only 3 datasets.
3. Performance of NOPRUNING for CLL is quite poor (average rank 4.60), especially on datasets with a high number of classes. This contrasts with our conclusions for AUC.
4. MDL and BIC have poor performance on datasets with a high number of classes. However, on most datasets with a low number of classes BIC has performance com-

	RAND	MDL	BIC	CHI	EBP	NOPRUNING
australian credit (2)	0.502	0.501	0.513	0.523	0.549	0.647 ◦
biodegradability (2)	0.880	0.886	0.892	0.891	0.907	0.925
breast wisconsin (2)	0.226	0.261	0.214	0.223	0.239	0.225
chess kr-kp (2)	0.046	0.068 ◦	0.045	0.047	0.048	0.036
diabetes (2)	0.766	0.769	0.779	0.764	0.820	0.916 ◦
german credit (2)	0.806	0.806	0.858	0.841	0.967 ◦	1.051 ◦
heart cleveland (2)	0.745	0.774	0.768	0.748	0.785	0.829
hiv (2)	0.184	0.197 ◦	0.188 ◦	0.189 ◦	0.200 ◦	0.208 ◦
ionosphere (2)	0.427	0.404	0.438	0.450	0.427	0.448
mushroom (2)	0.002	0.004	0.002	0.002	0.003	0.002
mutagenesis (2)	0.854	0.833	0.856	0.827	0.882	0.821
voting (2)	0.214	0.222	0.212	0.211	0.222	0.217
balance scale (3)	0.922	0.962	0.956	0.954	0.898	0.789 ●
splice (3)	0.282	0.282	0.276	0.288	0.305 ◦	0.337 ◦
thyroid (4)	0.068	0.084 ◦	0.077	0.068	0.073	0.091 ◦
vehicle (4)	0.941	1.071 ◦	0.980	0.952	1.070 ◦	1.161 ◦
annealing (5)	0.579	0.662 ◦	0.651 ◦	0.602	0.599	0.725 ◦
segment (7)	0.293	0.385 ◦	0.360 ◦	0.307	0.304	0.318 ◦
cora (10)	1.303	1.435 ◦	1.398 ◦	1.337	1.368	1.600 ◦
gene (10)	1.692	1.950 ◦	1.870 ◦	1.704	1.789 ◦	1.878 ◦
pen digits (10)	0.388	0.595 ◦	0.565 ◦	0.411 ◦	0.390	0.411 ◦
yeast (10)	1.759	1.778	1.779	1.716	1.878 ◦	2.426 ◦
vowel (11)	1.684	2.232 ◦	2.152 ◦	1.828 ◦	1.676	1.693
soybean (15)	1.453	1.901 ◦	1.892 ◦	1.510	1.562 ◦	1.736 ◦
primary tumor (21)	3.266	3.649 ◦	3.533 ◦	3.207	3.492 ◦	3.778 ◦
diterpenes (23)	2.010	2.261 ◦	2.217 ◦	1.989	2.013	2.221 ◦
average	0.857	0.96	0.941	0.869	0.903	0.98
wins/ties/losses		4/0/22	4/1/21	7/1/18	2/0/24	4/1/21
significant w/t/l		0/13/13	0/16/10	0/23/3	0/18/8	1/9/16
average rank	1.87	4.46	3.60	2.63	3.85	4.60

Table 4: Negative normalised Conditional Log-Likelihood for the six postpruning criteria (lower is better).

parable to RAND. The latter also holds for MDL but to a smaller extent (on fewer datasets). These conclusions are in line with the conclusions for AUC.

For MDL and BIC the conclusions in terms of CLL are very similar to those in terms of AUC. Concretely, the datasets on which RAND significantly outperforms MDL and BIC for CLL are to a large extent the same as the datasets on which this is the case for AUC. Moreover, these datasets almost all have a high number of classes. As discussed before, these are also the datasets on which the trees for MDL and BIC are the smallest. Hence, the results for CLL strengthen our conclusion that on datasets with a high number of classes MDL and BIC learn trees that are too small to perform well.

For NOPRUNING and EBP the conclusions in terms of CLL are quite different from those in terms of AUC. While NOPRUNING performed well for AUC, it performs quite poorly for CLL. For AUC NOPRUNING was significantly outperformed by RAND on only 5 datasets, for CLL this is the case on the same 5 datasets and on 11 additional datasets (mainly dataset with a high number of classes). Similar remarks apply to EBP. For AUC EBP was significantly outperformed by RAND on only 1 dataset, for CLL this is the case on the same dataset and on 7 additional datasets.

The fact that NOPRUNING and EBP perform well in terms of AUC but poorly in terms of CLL can be explained by poor calibration of their probability estimates. Probability estimates are called well-calibrated if they correspond to fractions in the data (Zadrozny and Elkan, 2001). Calibration of probability estimates on a given dataset typically decreases if tree size increases since probabilities are then estimated from fewer examples and tend to get more extreme (closer to 0 or 1). Since NOPRUNING and EBP have the largest trees of all criteria, they suffer from poor calibration more than the other criteria. Also, calibration typically depends on the number of classes since for a high number of classes the number of examples per class might be low, leading to poor calibration. This is consistent with the observation that NOPRUNING has poor CLL mainly on datasets with a high number of classes. Note that the fact that NOPRUNING and EBP perform bad for CLL but relatively well for AUC is explained by the fact that CLL is sensitive to calibration but AUC is not, see Section 5.1.2.

Several techniques have been proposed to improve calibration for large probability trees, such as curtailment (Zadrozny and Elkan, 2001), shrinkage (Wang and Zhang, 2006), m-branch (Ferri et al., 2003b) and others (Ling and Yan, 2003). We did not apply any of these recalibration techniques in this work for two reasons. First, investigating the influence of recalibration on the performance of different pruning criteria is beyond the scope of this paper and is more suited as the topic of a separate study. The reason for this is that it is not clear which recalibration technique should be incorporated since it is unknown which of these techniques perform best and since 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 the conclusions. Second, while applying some recalibration technique in our tree learner might indeed somewhat improve CLL for NOPRUNING and EBP, it is unlikely to change our conclusions. Concretely, recalibration might decrease the gap in CLL between RAND on the one hand and NOPRUNING and EBP on the other hand, but we strongly expect that

RAND would still be better than NOPRUNING and EBP. We expect this because for AUC, which is insensitive to calibration, RAND is also better than NOPRUNING and EBP.

5.3.4 INFLUENCE OF THE NUMBER OF CLASSES

In the previous sections we observed that the relative performance of the different pruning criteria depends on the number of classes of the dataset. In specific, we observed that BIC and (to a smaller extent) MDL have good performance if the number of classes is low but poor performance otherwise. The complexity penalties for MDL and BIC depend linearly on the number of classes. Hence, we hypothesised that this dependency is in a sense too strong, leading to too simple models with poor performance if the number of classes is high. In this section we discuss an experiment to verify this hypothesis.

We performed an experiment on the diterpenes dataset in which we varied the number of classes from 3 to 23 (the original diterpenes dataset has 23 classes, the highest number of all our datasets). For an experiment with C classes we considered only examples having one of the C most frequent classes and we computed the class distribution in the original dataset taking into account only these examples. We then sampled examples from the original dataset (without replacement) according to this class distribution. For all values of C we sampled the same total number of examples in order to eliminate any influence of a varying number of examples.⁵ For each value of C we performed 10 runs of 5-fold cross-validation with new samples in every run.

Figure 4 shows the results for tree size, AUC and “relative CLL”. Plotting CLL for each criterion directly does not give a clear view on the relative performance of the various pruning criteria since varying the number of classes for a given criterion changes CLL much more than varying the criterion for a given number of classes (CLL decreases as the number of classes increases). Hence, instead we plot for each pruning criterion “relative CLL”, the ratio of the CLL of the criterion to the CLL of RAND. A ratio greater than 1 for a certain criterion indicates that the criterion is worse than RAND, a ratio smaller than 1 indicates that the criterion is better than RAND.

We can see that for an increasing number of classes, tree size is roughly constant for RAND, EBP and NOPRUNING (the latter is not in the graph but constant at around 100 nodes). However, for CHI, MDL and BIC, tree size steadily decreases. For the highest number of classes in the graph, trees for MDL and BIC are extremely small, on average 2 internal nodes. These results confirm our hypothesis that the linear dependency of the complexity penalties on the number of classes causes MDL and BIC to learn very small trees if the number of classes is high.

When varying the number of classes, we are interested in the evolution of the relative differences in performance between all the criteria rather than in the evolution of the absolute performance of a single criterion. We can see that differences in AUC for CHI, RAND, EBP and NOPRUNING are roughly insensitive to the number of classes. However, the differences between AUC of these four criteria and that of MDL and BIC are heavily influenced by the number of classes. For 3 classes, MDL and BIC still have AUC almost as high as

5. We use 1157 examples. This is the number of examples having one of the three most frequent classes in the original dataset. Hence, for $C = 3$ all examples having one of the three classes are selected and no sampling is needed.

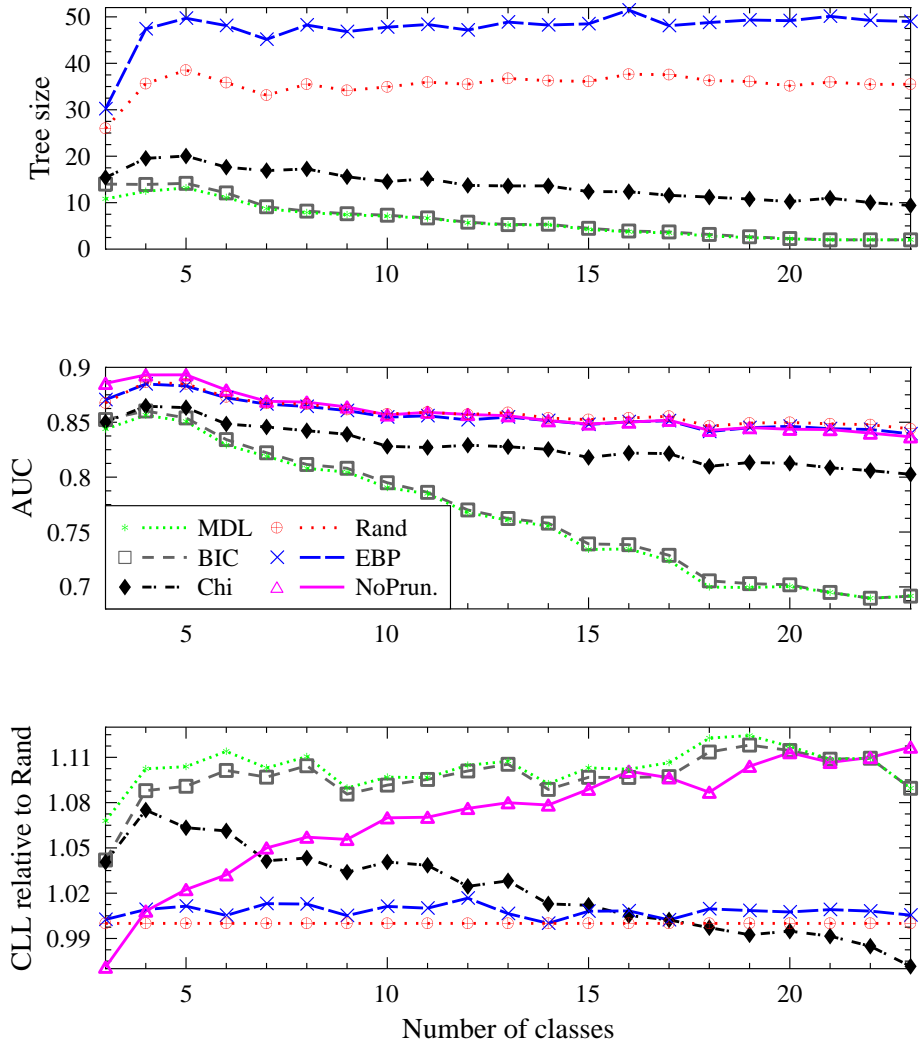


Figure 4: Influence of the number of classes for the diterpenes dataset. We show tree size (top), AUC (middle), and CLL relative to RAND (bottom). For tree size the results of NOPRUNING fall outside of the graph (after an initial increase when going from 3 to 4 classes, tree size for NOPRUNING stayed roughly constant at around 100 nodes).

	RAND	MDL	BIC	CHI	EBP	NOPRUNING
average	80.2	75.1	76.0	78.7	80.9	79.7
wins/ties/losses		5/0/21	8/1/17	7/1/18	19/0/7	9/1/16
significant w/t/l		0/15/11	0/17/9	0/20/6	3/23/0	2/19/5
average rank	2.90	4.69	3.87	3.52	2.08	3.94

Table 5: Summary of the results for accuracy (%) for the six postpruning criteria.

CHI. When increasing the number of classes, however, the gap between MDL and BIC on the one hand and the other criteria on the other hand increases steadily and MDL and BIC clearly become inferior.

From the graph for “relative CLL” we can see that EBP has a CLL close to that of RAND and the difference between the two is not heavily influenced by a varying number of classes. More interesting, MDL and BIC have poorer CLL than RAND and the gap initially increases with increasing number of classes. Also for NOPRUNING the gap with RAND increases with increasing number of classes. This can be explained by the fact that the higher the number of classes, the more NOPRUNING suffers from poor calibration. The trend for CHI (getting better relative to RAND with increasing number of classes) is difficult to explain.

The above observations about AUC and (to a smaller extent) CLL confirm our hypothesis that the decreasing tree size of MDL and BIC for an increasing number of classes indeed leads to worse performance. This suggests that the linear dependency of their complexity penalties on the number of classes is in a sense too strong.

5.3.5 CLASSIFICATION ACCURACY

The results for classification accuracy are given in Table 5. We only show a summary since the focus of this paper is on probability estimation rather than on classification. Full results are given in the online appendix (Fierens et al., 2007).

The conclusions for classification accuracy are slightly different than for AUC and CLL. The best criterion is no longer RAND (average rank 2.90) but EBP (average rank 2.08). This is not surprising since EBP was actually designed to maximize classification accuracy and is the most commonly used criterion for classification purposes. Taking this into account it is interesting that EBP only significantly outperforms RAND on 3 out of 26 datasets. Note also that MDL again has the lowest performance of all criteria (average rank 4.69).

5.3.6 RUNNING TIME

Table 6 shows a summary of the results for running time (we use relative running time with respect to RAND). Full results are given in the online appendix (Fierens et al., 2007).

The main observations are the following.

1. Running times for CHI, EBP and NOPRUNING are almost equal.
2. Running times for MDL and BIC are slightly smaller than for CHI, EBP and NOPRUNING

	absolute time for RAND	relative time for MDL	relative time for BIC	relative time for CHI	relative time for EBP	relative time for NOPRUNING
min	0.9s	0.03	0.03	0.08	0.08	0.08
max	146.7s	0.22	0.38	0.83	0.83	0.80
average	23.0s	0.12	0.17	0.37	0.36	0.36
min	14.2s	0.23	0.23	0.39	0.39	0.39
max	1693.1s	0.59	0.74	0.84	0.86	0.81
average	327.3s	0.34	0.42	0.63	0.62	0.61

Table 6: Summary of running times for the six postpruning criteria. The upper half of the table is for attribute-value datasets, the lower half for relational datasets. Minimum, maximum and average running times were computed over the respective datasets. For RAND we show absolute running times in seconds, for the other criteria we show relative running times with respect to RAND.

- Running times for RAND are the highest of all criteria. On attribute-value datasets the time for the other criteria is on average between a fraction 0.12 and 0.36 of the time for RAND, on relational datasets it is between a fraction of 0.34 and 0.61.

The running time for a postpruning criterion is composed of two parts: the time to build the unpruned tree and the time to do the actual postpruning. For all postpruning criteria considered, the actual postpruning time is negligible as compared to the time for building the unpruned tree. This is because all information needed for postpruning (entropy in nodes, description length of nodes, χ^2 -scores, randomizations, ...) can already be computed and stored during the construction of the unpruned tree. Hence, we do not need to access the dataset anymore during the actual postpruning, which makes this step very efficient.

Taking the above into account, we can now explain our observations. The time to build an unpruned tree is slightly smaller for MDL and BIC than for CHI, EBP and NOPRUNING since for MDL and BIC we use an optimization that allows us to stop building the unpruned tree earlier, see Sections 4.3 and 4.4. The time to build an unpruned tree is almost equal for CHI, EBP and NOPRUNING since in all these cases we have to build the full unpruned tree and for each candidate test in each node only simple computations have to be performed. The time to build an unpruned tree for RAND is the largest since we have to build the full unpruned tree and for each candidate test in each node a heuristic value (information gain) needs to be computed N_{rand} times. However, this does not mean that RAND is a factor N_{rand} (=100) slower than the other criteria since each candidate test still has to be executed only once on each example, see Section 4.2. Hence, the exact computational overhead for RAND depends on the ratio of the time needed to compute the results of candidate tests on the examples over the time to compute information gains given these results. For relational datasets, candidate tests are usually more expensive than for attribute-value datasets due to issues such as aggregates (Van Assche et al., 2006) and lookahead (Blockeel and De Raedt, 1997). Hence, the overhead of RAND relative to the other criteria is smaller on relational datasets than on attribute-value datasets. This can also be seen in Table 6.

We conclude that randomization is obviously the most expensive criterion but the overhead relative to the other criteria is smaller than one might expect, both on attribute-value datasets and on relational datasets.

5.3.7 OVERALL CONCLUSIONS

The main conclusion from our experiments is that RAND is the best criterion to optimize the quality of the probability estimates in terms of AUC and CLL. RAND regularly significantly outperforms the other criteria. In contrast, there is only one criterion that sometimes significantly outperforms RAND, namely NOPRUNING, but this happens very rarely. Additionally, NOPRUNING has the drawback that its trees are typically huge (on average a factor 5.6 larger than for RAND). The drawback of RAND is that it has a higher running time than the other criteria.

Based on these observations, our recommendation is to use RAND whenever its running time is not prohibitive and to use some alternative otherwise. In general the best alternative is CHI, which has trees of comparable size and also performs relatively well for AUC and CLL. In specific scenarios other alternatives exist as well. On datasets with a low number of classes, BIC typically performs quite well. Also, if tree size is not important and the goal is to maximize AUC (rather than CLL), NOPRUNING is a good option too.

6. Related Work

In Section 1 we already briefly discussed the main previous works on pruning criteria for probability trees (Provost and Domingos, 2003; Ferri et al., 2003b; Neville et al., 2003; Friedman and 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 in more detail. First, randomization is the best criterion in our experiments and hence deserves most attention. Second, while randomization has been used before for pruning decision trees, most of these previous approaches have some subtle yet important differences with our approach.

Frank and Witten (1998) use randomization tests for prepruning. They use randomization to compute the acceptability for each candidate test individually but do not use Bonferroni correction to take into account the influence of the number of candidate tests. This has the disadvantage of potential overfitting when there are many candidate tests (the more tests, the more likely that at least one of them looks acceptable, even if they are all uncorrelated to the class variable). Note that in our CHI criterion we also compute the acceptability for each candidate test individually but we do apply Bonferroni correction, which assumes that all tests are mutually independent. Also note that in our RAND criterion we compute the acceptability of the best candidate test (instead of the acceptability of each candidate test individually), which makes Bonferroni correction unnecessary. This has the advantage of automatically taking into account the influence not only of the number of candidate tests but also of any potential mutual dependencies between candidate tests.

Neville et al. (2003) use randomization tests for prepruning relational probability trees. The main difference between their approach and our RAND criterion is that they randomize attribute values whereas we randomize class labels. Both approaches are equivalent on

attribute-value data but the approach of Neville et al. has the advantage that it corrects certain kinds of bias that often occur in relational data, see Jensen et al. (2003) for details. The disadvantage of their approach is that it is computationally more expensive than our RAND criterion. Whereas we have to compute the result of each candidate test on each example only once, Neville et al. have to compute it N_{rand} times with N_{rand} the number of randomizations (since randomizing attribute values changes these results, while randomizing class labels does not). Hence, the running time for their approach is N_{rand} times the running time without randomization, whereas the running time for our RAND criterion is a lot smaller (see Sections 4.2 and 5.3.6).⁶ This is the reason why we did not use their approach in our work. Recall that our main conclusion from the experiments is that randomization gives the best rankings and probability estimates of all pruning criteria. Applying the approach of Neville et al. to our work (in a postpruning version) could improve the results of randomization on the relational datasets and hence could make our conclusion even stronger.

Oates and Jensen (1998) use randomization tests for subtree postpruning. 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 N_{rand} 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 sampling distribution. The advantage of the approach of Oates and Jensen is that it might give slightly better results than our RAND postpruning criterion since they use subtree postpruning whereas we use substump postpruning. Hence, again it holds that applying this approach to our work could make our conclusion about the superiority of randomization even stronger. The disadvantage of the approach of Oates and Jensen is that it is computationally expensive. When an internal node is considered for pruning, for each randomization a new subtree needs to be build on the randomized data. Hence, the running time is N_{rand} times the running time without randomization (again the authors do not report running times). This is not the case for our RAND criterion.

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, this paper is the first to show that randomization tests for decision tree pruning are computationally feasible in practice. Concretely, although the running time of our RAND criterion is in theory linear in the number of randomizations N_{rand} , in practice the computational overhead with respect to other pruning criteria is a lot smaller than a factor N_{rand} .

7. Conclusion

From the current state of the literature there is no clear view on the relative performance of different pruning criteria for probability trees and hence it is unclear which criteria are preferable under which circumstances. In this paper we surveyed six of the most important

6. Neville et al. (2003) do not discuss the computational complexity or running times of their algorithm, but we do not see how this can be made more efficient.

pruning criteria for probability trees and discussed their theoretical advantages and disadvantages. We also performed an extensive experimental study of their relative performance.

Our main conclusion is that a postpruning criterion based on randomization tests gives the best probability estimates and rankings and also learns relatively small trees. Our recommendation is to use this criterion whenever its running time is not prohibitive. Otherwise, the best alternative is a chi-square based criterion, which learns trees of comparable size and also performs relatively well. In specific scenarios other alternatives exist as well. On datasets with a low number of classes, the Bayesian Information Criterion typically performs quite well. Also, if tree size is not important and the goal is to obtain good rankings (rather than good probability estimates directly), learning unpruned trees is a good option too.

Acknowledgments

Daan Fierens is supported by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT Vlaanderen). Jan Ramon and Hendrik Blockeel are postdoctoral fellows of the Fund for Scientific Research (FWO) of Flanders. This research is also supported by the Project GOA 2003/8 *Inductive Knowledge Bases* and the FWO Project *Probabilistic-Logical Learning*.

Appendix A.

In this appendix we show how to speed up the subtree postpruning algorithm for the MDL and BIC criteria.

In the first step of postpruning we have to build an unpruned tree. We can speed-up this step slightly since sometimes we can derive from the properties of the MDL or BIC criterion that a subtree will definitely be pruned in the second step, even before actually building this subtree in the first step. Note that this optimization does not change the final result.

First we discuss MDL. Consider a node l that is leaf at some point during the construction of the unpruned tree. If we continue the tree building from l by replacing l with a subtree, then $DL(\mathcal{T})$ will increase and $DL(\mathcal{D} | \mathcal{T})$ will decrease. The minimum increase of $DL(\mathcal{T})$ is $2 + \log_2(N_{test}) + 0.5(C - 1)\log_2(N)$, since this is the increase when l is replaced by the smallest subtree possible, a stump. The maximum decrease of $DL(\mathcal{D} | \mathcal{T})$ is $N_l H(\mathcal{D}_l)$. To see this, note that the contribution of a leaf l to $DL(\mathcal{D} | \mathcal{T})$ is $N_l H(\mathcal{D}_l)$ and if l is replaced by a subtree the contribution of this subtree to $DL(\mathcal{D} | \mathcal{T})$ is at least 0 (it is 0 when all leaves of the subtree would be pure and greater than 0 otherwise). If the minimum increase of $DL(\mathcal{T})$ due to replacing l with a subtree is greater than the maximum decrease of $DL(\mathcal{D} | \mathcal{T})$ we know even without building this subtree first that it will be pruned in the second step. Hence, whenever for a node l the condition $N_l H(\mathcal{D}_l) < 2 + \log_2(N_{test}) + 0.5(C - 1)\log_2(N)$ occurs in the first step, we simply stop the tree building along that branch and create a leaf. As far as we know, this optimization has not been described in the literature before.

For BIC a similar optimization can be used: whenever $N_l H(\mathcal{D}_l) < 0.5(C - 1)\log_2(N)$ occurs in the first step for a node l we stop the tree building along that branch and create a leaf.

References

- H. Blockeel and L. De Raedt. Lookahead and discretization in ILP. In *Proceedings of the Seventh International Workshop on Inductive Logic Programming*, volume 1297 of *Lecture Notes in Artificial Intelligence*, pages 77–85. Springer-Verlag, 1997.
- H. Blockeel and L. De Raedt. Top-down induction of first order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, June 1998.
- R.R. Bouckaert and E. Frank. Estimating replicability of classifier learning experiments. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.
- A.P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30:1145–1159, 1997.
- D.M. Chickering and D. Heckerman. Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning*, 29(2-3):181–212, 1997.
- P.R. Cohen and D. Jensen. Overfitting explained. In *Preliminary Papers of the Sixth International Workshop on Artificial Intelligence and Statistics*, pages 115–122, 1997.
- P. Domingos. Occam’s two razors: The sharp and the blunt. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, pages 37–43, 1998.
- J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In A. Prieditis and S. Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 194–202. Morgan Kaufmann, 1995.
- S. Džeroski and N. Lavrač. *Relational Data Mining*. Springer-Verlag, 2001.
- F. Esposito, D. Malerba, and G. Semeraro. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):476–491, 1997.
- T. Fawcett. Using rule sets to maximize ROC performance. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 131–138, 2001.
- C. Ferri, P. Flach, and J. Hernandez-Orallo. Decision trees for ranking: Effect of new smoothing methods, new splitting criteria and simple pruning methods. Technical report, 2003a.
- C. Ferri, P. Flach, and J. Hernandez-Orallo. Improving the AUC of Probabilistic Estimation Trees. In *Proceedings of 14th European Conference on Machine Learning, Porto, Portugal*, 2003b.

- C. Ferri, J. Hernández-Orallo, and M.A. Salido. Volume under the ROC Surface for multi-class problems. In *Proceedings of 14th European Conference on Machine Learning*, pages 108–120, 2003c.
- D. Fierens, J. Ramon, H. Blockeel, and M. Bruynooghe. Online appendix to this article, 2007. <http://www.cs.kuleuven.be/~daanf/pruning>.
- D. Fierens, H. Blockeel, M. Bruynooghe, and J. Ramon. Logical Bayesian networks and their relation to other probabilistic logical models. In *Proceedings of the 15th International Conference on Inductive Logic Programming*, volume 3625 of *Lecture Notes in Computer Science*, pages 121–135. Springer, 2005a.
- D. Fierens, J. Ramon, H. Blockeel, and M. Bruynooghe. A comparison of approaches for learning probability trees. In *Proceedings of 16th European Conference on Machine Learning, Porto, Portugal*, volume 3720 of *Lecture Notes in Artificial Intelligence*, pages 556–563, 2005b.
- D. Fierens, J. Ramon, H. Blockeel, and M. Bruynooghe. A comparison of approaches for learning probability trees. Technical Report CW 418, Department of Computer Science, Katholieke Universiteit Leuven, June 2005c. <http://www.cs.kuleuven.ac.be/publicaties/rapporten/cw/CW418.abs.html>.
- E. Frank and I.H. Witten. Using a permutation test for attribute selection in decision trees. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)*, pages 152–160. Morgan Kaufmann, 1998.
- N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.
- N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. In *Proceedings of the NATO Advanced Study Institute on Learning in graphical models*, pages 421–459, Norwell, MA, USA, 1998. Kluwer Academic Publishers.
- L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In S. Džeroski and N. Lavrač, editors, *Relational Data Mining*, pages 307–334. Springer-Verlag, 2001.
- D. Grossman and P. Domingos. Learning Bayesian network classifiers by maximizing conditional likelihood. In *Proceedings of 21st International Conference on Machine learning (ICML04)*, 2004.
- T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.
- D. Heckerman, D. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.
- D. Jensen, J. Neville, and M. Rattigan. Randomization tests for relational learning. Technical Report 03-05, Department of Computer Science, University of Massachusetts, 2003.

- D. Jensen and M. Schmill. Adjusting for multiple comparisons in decision tree pruning. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, 1997.
- D. Jensen and P.R. Chohen. Multiple comparisons in induction algorithms. *Machine Learning*, 38(3):309–338, 2000.
- K. Kersting and L. De Raedt. Bayesian logic programming: Theory and tool. In *An Introduction to Statistical Relational Learning*. MIT Press, 2007.
- S. Kramer, L. De Raedt, and C. Helma. Molecular feature mining in HIV data. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-01)*, pages 136–143. ACM Press, 2001.
- C.X. Ling and R.J. Yan. Decision tree with better ranking. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, pages 480–487, 2003.
- A. McCallum, K. Nigam, J. Rennie, and K. Seymore. A machine learning approach to building domain-specific search engines. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI99)*, pages 662–667, 1999.
- M. Mehta, J. Rissanen, and R. Agrawal. MDL-based decision tree pruning. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 216–221, 1995.
- C.J. Merz and P.M. Murphy. UCI repository of machine learning databases [<http://www.ics.uci.edu/~mlearn/mlrepository.html>], 1996. Irvine, CA: University of California, Department of Information and Computer Science.
- J. Neville, D. Jensen, L. Friedland, and M. Hay. Learning relational probability trees. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.
- T. Oates and D. Jensen. Large datasets lead to overly complex models: an explanation and a Solution. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, 1998.
- F. Provost and P. Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52:199–216, 2003.
- J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann series in Machine Learning. Morgan Kaufmann, 1993.
- J.R. Quinlan and R.L. Rivest. Inferring decision trees using the Minimum Description Length principle. *Information and Computation*, 80:227–248, 1989.
- J. Ramon, T. Croonenborghs, D. Fierens, H. Blockeel, and M. Bruynooghe. Generalized ordering-search for learning directed probabilistic logical models. *Machine Learning, special issue on Inductive Logic Programming*, 2007. Conditionally accepted.

- J. Ramon, D. Fierens, F. Guiza, H. Blockeel, G. Meyfroid, M. Bruynooghe, and G. Van Den Berghe. Mining data from intensive care patients. *Advanced Engineering Informatics, Special Journal Issue on Applications Eligible for Data Mining*, 2006. In press.
- G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.
- A. Van Assche, C. Vens, H. Blockeel, and S. Džeroski. First order random forests: Learning relational classifiers with complex aggregates. *Machine Learning*, 2006.
- C.S. Wallace and J.D. Patrick. Coding decision trees. *Machine Learning*, 11:7–22, 1993.
- B. Wang and H. Zhang. Improving the ranking performance of decision trees. In *Proceedings of the 17th European Conference on Machine Learning*, pages 461–472, 2006.
- B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In *Proceedings of ICML-2001 - Eighteenth International Conference on Machine Learning*, pages 609–616. Morgan Kaufmann, 2001.