Mach Learn (2008) 71: 55–88
DOI 10.1007/s10994-007-5044-4

A k-norm pruning algorithm for decision tree classifiers based on error rate estimation

Mingyu Zhong · Michael Georgiopoulos · Georgios C. Anagnostopoulos

Received: 27 February 2007 / Revised: 10 December 2007 / Accepted: 12 December 2007 /
Published online: 4 January 2008

Abstract Decision trees are well-known and established models for classification and regression. In this paper, we focus on the estimation and the minimization of the misclassification rate of decision tree classifiers. We apply Lidstone's Law of Succession for the estimation of the class probabilities and error rates. In our work, we take into account not only the expected values of the error rate, which has been the norm in existing research, but also the corresponding reliability (measured by standard deviations) of the error rate. Based on this estimation, we propose an efficient pruning algorithm, called k-norm pruning, that has a clear theoretical interpretation, is easily implemented, and does not require a validation set. Our experiments show that our proposed pruning algorithm produces accurate trees quickly, and compares very favorably with two other well-known pruning algorithms, CCP of CART and EBP of C4.5.

Keywords Decision tree · Pruning · Law of succession

1 Introduction

Decision trees have been among the most wide-spread models in machine learning. Their advantages include the interpretability of the model and the capability to handle both numerical attributes and categorical attributes even with missing values. Like other approaches that

Editor: Hendrik Blockeel.

M. Zhong · M. Georgiopoulos (✉)
School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, USA
e-mail: michaelg@mail.ucf.edu

M. Zhong
e-mail: myzhong@ucf.edu

G.C. Anagnostopoulos
Department of Electrical & Computer Engineering, Florida Institute of Technology, Melbourne, FL 32901, USA
e-mail: georgio@fit.edu

construct their models from data, a decision tree classifier over-adapted to the training set tends to generalize poorly, when it is confronted with unseen instances. Therefore, it has been widely accepted that the grown tree should be pruned.

A number of pruning algorithms have been proposed. They include Minimal Cost-Complexity Pruning (CCP) of CART (Classification and Regression Trees; Breiman et al. 1984, p. 66), Error Based Pruning (EBP) of C4.5 (Quinlan 1993, p. 37), Minimum Error Pruning (MEP) (Niblett and Bratko 1986; Cestnik and Bratko 1991), Reduced Error Pruning (REP), Pessimistic Error Pruning (PEP) (Quinlan 1999, for both REP and PEP), MDL-Based Pruning (Mehta et al. 1995), Classifiability Based Pruning (Dong and Kothari 2001), pruning using Backpropagation Neural Networks (Kijsirikul and Chongkasemwongse 2001). Some of the pruning algorithms are briefly analyzed and empirically compared in Esposito et al. (1997).

Through the analysis conducted in this paper we introduce a new pruning algorithm, referred to as *k-norm pruning algorithm* that features the following properties: it has a clear theoretical interpretation based on error rate estimation and minimization, it does not require cross-validation or a separate validation set, it can find the optimal pruned tree within one traversal of the tree, and, finally, it is simple and easy to implement. Furthermore, it compares favorably with two well known pruning strategies, CCP and EBP, as well as other pruning algorithms that have appeared in the decision tree literature.

The rest of this paper is organized as follows. In Sect. 2, we first present an overview of some of the well-known misclassification rate estimation approaches, and this discussion serves as a motivation for our estimation approach. Then, the necessary background knowledge is covered in Sect. 3, which introduces the reader to decision trees and probability estimation. We elaborate on the theoretical analysis of the tree predictions in Sect. 4. In particular, we apply Lidstone's Law of Succession (LLS; Lidstone 1920) for the estimation of averages and standard deviations of misclassification rates in decision tree classifiers. Based on our analysis, we introduce the appropriate equations to estimate the tree's prediction accuracy and a pruning algorithm with the properties mentioned earlier. In Sect. 5, we show the experimental results of our pruning algorithm by comparing it to two other classical algorithms, as well as other pruning algorithms. Furthermore, in Sect. 5, we emphasize the advantages of the k-norm approach compared to alternative pruning methods, such as CCP and EBP. We summarize our work in Sect. 6. In the appendices, we provided a detailed example of our estimation approach to a well studied problem, as well as proofs of our theorems.

2 Motivation of our work

It would be very easy to estimate the error rate on unseen data if a separate validation set with sufficient size were given. When only the training set is given, one can separate the training set into two parts, one for growing the tree and one for validating the tree (see REP in Quinlan 1999 and its two successors in Kääriäinen and Elomaa 2003; Kääriäinen et al. 2004 for example). In this case, however, the resulting tree does not utilize all the training examples in its design. A better solution is to use a V-fold cross-validation (Stone 1978), which has been successfully applied in CCP. Nevertheless, cross-validation is usually a computationally expensive procedure. Most of the estimation approaches without validation can be divided into two categories: Maximum Likelihood Estimation and Posterior (Bayesian) Estimation. They are discussed in the next two subsections.

2.1 Maximum likelihood estimation

This approach seems to have attracted most of the research interest in error rate estimation for decision tree classifiers. Although the maximum likelihood estimate cannot be used by itself (otherwise the fully grown tree would have the highest estimated accuracy), it can provide some useful information accompanied by Hoeffding's Inequality (Hoeffding 1963), Chernoff's Inequality (Chernoff 1952), or the Binomial confidence interval. Many researchers have experimented with different confidence levels and/or applying other inequalities to achieve better confidence bounds/intervals for the generalization accuracy (Quinlan 1993; Mansour 1997; Freund 1998; Kearns and Mansour 1998; Mansour and McAllester 2000; Kääriäinen and Elomaa 2003). Note that in EBP, the equation for computing the estimated error rate is not explicitly given; it can be found in Windeatt and Ardeshir (2001) and turns out to be a Bernoulli one-sided Confidence Interval.

The above approaches are very straightforward. One of the most important advantages is that they make no assumption on the distribution of the error rate. Some tree pruning algorithms have been proposed that can finish within one traversal of the tree based on the upper bound of the error rate. In applying these approaches, however, many researchers have sidestepped an important issue related to the correct interpretation of Hoeffding's/Chernoff's Inequalities, as pointed out recently in Vardeman and Jobe (2001, p. 342) it is not safe to state that the generalization accuracy/error rate (expected error rate) has a specific interval/bound at a certain probability by simply relying on the maximum likelihood estimates of this error rate from a single training data set. Of course, we could still use the resulting intervals/bounds as heuristics to estimate the unknown accuracy, but we must keep in mind the limitations of such an estimate, as explained above.

To the best of our knowledge, we have not seen any statistically significant experimental results (at least 1000 points in the database) showing that the above approaches can yield a pruning algorithm superior to those using validation or those using posterior estimations. For example, EBP tends to under-prune a tree as shown in Esposito et al. (1997). The behavior of EBP can be explained by the theoretical analysis in Elomaa and Kääriäinen (2001), where the authors proved that in EBP, a decision node will not be pruned unless all its immediate children are either originally leaves or pruned to leaves.

2.2 Posterior estimation

Contrary to maximum likelihood estimation, posterior estimates assume that unknown parameters $\boldsymbol{\theta}$ are random variables, and model them by a posterior joint *probability density function* (abbreviated as PDF; denoted by f) based on Bayesian theory:

$$\begin{aligned} f(\boldsymbol{\theta}|Observations) &= \frac{f(Observations|\boldsymbol{\theta})\, f(\boldsymbol{\theta})}{f(Observations)} \\ &= \frac{f(Observations|\boldsymbol{\theta})\, f(\boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} f(Observations|\boldsymbol{\theta})\, f(\boldsymbol{\theta}) d\boldsymbol{\theta}}. \end{aligned} \tag{1}$$

When $\boldsymbol{\theta}$ stands for one or more probabilities and the observations are Bernoulli trials, $f(Observations|\boldsymbol{\theta})$ is not difficult to compute. The apparent difficulty is the unknown prior PDF $f(\boldsymbol{\theta})$. An explicit expression of this prior PDF must be assumed, and its properness is usually difficult to verify. Nevertheless, this approach allows, at least, the production of meaningful results.

A typical example is MEP (Niblett and Bratko 1986), where LLS (Lidstone 1920) is used (see Sect. 3.2 for details). LLS can be derived by choosing the prior joint PDF to be represented by the Dirichlet distribution (see (14)) and by choosing the posterior expected value as the estimated value. In our paper, however, we will prove that MEP tends to under-prune a tree: under a loose condition, the estimated error rate always drops if a split reduces the number of training misclassifications by at least one, and thus will not be pruned (see Theorem 2 in Sect. 4.4). We argue that the expected value alone does not serve as a good estimate of the misclassification rate, because it does not provide us with any information about the reliability of the estimate (analogous to the confidence interval in maximum likelihood estimation). For instance, an estimate in the range of 0.2 ± 0.01 is much better than an estimate in the range of 0.2 ± 0.2 although they have same expected values (see Sect. 4.3 for a quantitative example).

2.3 k-norm estimate

In our work we choose to prune a tree by minimizing the estimated error rate, because we desire not only the optimal pruned tree but also the performance prediction. We choose not to rely on a validation set because validation may not be practical when the training set is small and it is expensive to use when the training set is large. To the best of our knowledge, there is no error estimation approach with a strong theoretical support that does not require validation. Based on the references discussed above that are related to prior tree pruning work, MEP appears to be the most promising approach for improvement: the posterior estimations can yield interpretable results as long as the prior PDF is selected properly. The Dirichlet distribution provides an explicit expression for the prior PDF and yields LLS, which has been widely accepted in the statistical literature. Since expected values are not good enough for error rate estimation, we also calculate the standard deviation of the error rate. These considerations lead us in a natural way to the k-norm error rate pruning approach, a generalization of the 2-norm error rate pruning that relies on estimation of averages and standard deviations of error rates (see Sect. 4.4 that provides a quantitative explanation of our motivation to rely on expected values and standard deviations of error rates to assess the tree pruning decisions).

3 Preliminaries

This section provides some basic information on decision trees and probability estimation approaches, needed for the rest of the paper.

3.1 Decision trees

A typical tree is shown in Fig. 1.[1] To train a tree classifier, usually two phases are required. One is the growing phase, where the tree is grown to a sufficient size, and the other is the pruning phase, where the tree is pruned back to prevent over-training (Breiman et al. 1984). In this paper, we do not focus on the growing phase but the pruning phase with the error rate estimation. To pave our way for error rate estimation, we first discuss tree classification from the perspective of a-posteriori probabilities.

[1] Note: The variables x_1 to x_4 represent the attributes, and y represents the variable whose value the tree will be predicting. The nodes of the tree are designated by the index t; the labels "Y" and "N" under each question stand for the answers "Yes" and "No", respectively, to the question.

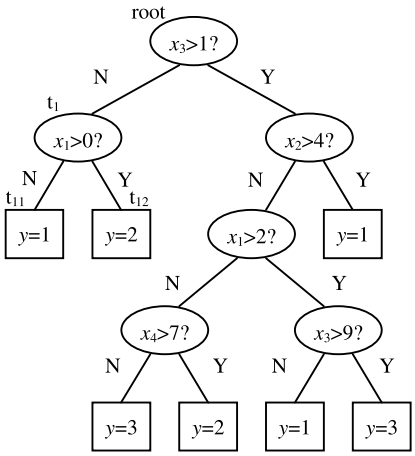


Fig. 1 The graph of a decision tree

3.1.1 Probabilistic view of tree classification

Let $\mathbf{X}$ denote an input instance that can be treated as a random vector, and let C_j represent the event that $\mathbf{X}$ belongs to the j-th class ($j = 1, 2, \ldots, J$, where J is the number of classes). To view the decision tree classification in a probabilistic way, we can express $P[C_j|\mathbf{X}]$ as

$$P[C_j|\mathbf{X}] = P[C_j|A_{root}, \mathbf{X}], \tag{2}$$

where *root* is the root of the tree, A_t is the event that node t is activated (receives the input $\mathbf{X}$). Since the root is always activated, A_{root} is always true. For other nodes, A_t is defined by the splits involving the ancestors of node t. For example, in Fig. 1, $A_{t_1} = (x_3 \leq 1)$ and $A_{t_{11}} = (x_3 \leq 1, x_1 \leq 0)$ (the comma in the parentheses stands for intersection of events). For most decision trees, we can rely on the following assumptions:

Partitioning Assumption: For any decision node d, the events $\{A_c|A_d, c \in \mathit{Children}(d)\}$ are mutually exclusive and collectively exhaustive, where $\mathit{Children}(d)$ is the set of immediate children of node d.

Determinacy Assumption: Given $\mathbf{X}$, the path of $\mathbf{X}$ from the root to a leaf is deterministic, that is, $P[A_c|A_d, \mathbf{X}]$ is either 0 or 1. For example, in Fig. 1, $P[A_{t_{11}}|A_{t_1}, \mathbf{X}] = I[x_1 \leq 0]$, where $I[\Phi]$ is the indicator function (1 if Φ is true and 0 otherwise).

Piece-wise Independence Assumption: At a leaf l, the class label is assumed to be independent of $\mathbf{X}$.

The Partitioning Assumption leads to the following equation: for any decision node d,

$$P[C_j|A_d, \mathbf{X}] = \sum_{c \in \mathit{Children}(d)} P[C_j|A_c, A_d, \mathbf{X}] P[A_c|A_d, \mathbf{X}]. \tag{3}$$

Note that a child of d is activated only if d is activated, that is, A_c implies A_d, or $(A_c, A_d) = A_c$. Therefore, (3) can be rewritten as

$$P[C_j|A_d, \mathbf{X}] = \sum_{c \in \mathit{Children}(d)} P[C_j|A_c, \mathbf{X}] P[A_c|A_d, \mathbf{X}]. \tag{4}$$

By recursion this telescopes to

$$P[C_j|\mathbf{X}] = \sum_{l \in Leaves} P[C_j|A_l, \mathbf{X}]P[A_l|\mathbf{X}]. \tag{5}$$

Under the Piece-wise Independence Assumption, for any leaf l,

$$P[C_j|A_l, \mathbf{X}] \approx P[C_j|A_l]. \tag{6}$$

$P[C_j|A_l]$ is unknown and has to be estimated (e.g., by Maximum Likelihood Estimation or LLS). Let $P^*[C_j|A_l]$ denote the estimated value of $P[C_j|A_l]$ (the asterisk representing the estimated value throughout this paper). The predicted class probability $P^*[C_j|\mathbf{X}]$ is estimated as follows:

$$P^*[C_j|\mathbf{X}] = \sum_{l \in Leaves} P^*[C_j|A_l]P[A_l|\mathbf{X}]. \tag{7}$$

Under the Determinacy Assumption, any $\mathbf{X}$ activates only one leaf (denoted by $l_{\mathbf{X}}$). Therefore,

$$P^*[C_j|\mathbf{X}] = P^*[C_j|A_{l_{\mathbf{X}}}]. \tag{8}$$

Interestingly, whether the Maximum Likelihood Estimation or LLS is applied for computing the estimated value $P^*[C_j|A_l]$,

$$Label^*(\mathbf{X}) = Label(l_{\mathbf{X}}) = \arg\max_j P^*[C_j|A_{l_{\mathbf{X}}}] = \arg\max_j n_{j,l_{\mathbf{X}}}, \tag{9}$$

where $n_{j,t}$ is the number of training examples of class j in a node t that receives $\mathbf{X}$ (this notation is also applicable to decision nodes).

The assumptions presented here are used for the proof of Theorem 1 and Corollary 1, based on which the k-norm estimation algorithm is derived (see Sect. 4.2 and Sect. 4.4). The equations, presented above, are used in the discussion that follows.

3.1.2 Discussion

There is a dilemma in tree induction regarding the under-training/over-training of the tree. The estimation in (7) includes two approximations:

$$P[C_j|\mathbf{X}] = \sum_{l \in Leaves} P[C_j|A_l, \mathbf{X}]P[A_l|\mathbf{X}] \approx \sum_l P[C_j|A_l]P[A_l|\mathbf{X}], \tag{10}$$

$$P[C_j|A_l] \approx P^*[C_j|A_l]. \tag{11}$$

As the tree keeps growing and each leaf l occupies a smaller region in the attribute space, the first approximation becomes more and more accurate, since $P[C_j|\mathbf{X}, A_l]$ of (10) is substituted by the piece-wise constant function $P[C_j|A_l]$. However, the estimation of $P[C_j|A_l]$ actually turns out to be less reliable since fewer training examples are passed to l, and it is the cause of why a fully grown tree usually exhibits poor generalization. The dilemma can never be completely overcome, even if the *Piece-wise Independence Assumption* is discarded and/or Lidstone's Estimation is applied, because when fewer examples are given, the approximation of $P[C_j|\mathbf{X}, A_l]$ using any predefined model (such as the Gaussian PDF or the uniform PDF) appears easier but is less reliable.

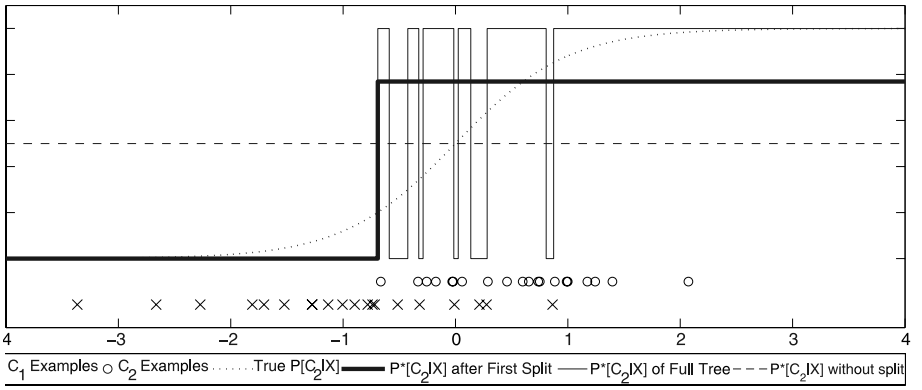


Fig. 2 Estimated Class probabilities obtained by CART for a 2-class Gaussian data-set. In the graph the true class probability is shown, as well as the class probabilities estimated after the first split and for the fully grown tree

An example is shown in Fig. 2, to illustrate our point that the estimation of class probabilities becomes less reliable as the tree grows in size. We generated an artificial Gaussian database with 2 classes (20 examples in each class) and one attribute X ($X|C_1 \sim \mathcal{N}(-1, 1)$, $X|C_2 \sim \mathcal{N}(1, 1)$; the two classes are plotted at different vertical locations only for visual purpose). We used CART to grow a tree. Before any split (when the tree has only one leaf), each estimated class probability has a constant value 0.5, giving no information for the classification. After the first split, the estimation of $P[C_j|\mathbf{X}]$ is improved significantly, especially for the region to the left of the split: the root-mean-square-error of the probability estimate decreases from 0.43 to 0.19, and the error rate decreases from 0.5 to 0.086. For the fully grown tree (where each leaf is pure), the estimated class probability has many spikes and it does not look better than the estimated class probability with only the first split; for the fully grown tree the root-mean-square-error of the probability estimate increases from 0.19 to 0.22, and the error rate increases from 0.086 to 0.091.

Since growing the tree to its maximum size might not be beneficial, there is a point at which we should stop growing the tree. Nevertheless, during the growing phase it is difficult to foresee how many more splits under a newly created leaf will be finally found and whether or not these splits will be useful. From this point of view, we also support the widely accepted approach of growing the tree to its maximum size and then pruning the tree, for it allows us to find all candidate pruned trees. Now, our main problems are (1) how to evaluate a candidate pruned tree, because the true PDF is seldom known, and (2) how to find the optimal pruned tree efficiently, as there are usually too many candidate pruned trees to choose from. These problems are addressed in Sect. 4.

3.2 Lidstone's law of succession

An important issue in decision trees, as well as many other classifiers, is the estimation of the class probabilities given a set of training instances. To be more general, let Φ_h ($h = 1, 2, \ldots, H$) be H *mutually exclusive and collectively exhaustive* events. Let p_h denote $P[\Phi_h]$ and N_h denote the number of occurrences of Φ_h in N independent trials. Obviously, N_h is a random variable. Suppose that in N independent trials we observed n_h occurrences of Φ_h (an example would be observing n_j out of N training instances that belong to class j). Our task here is to estimate $p_1, p_2, \ldots, p_H$ given $n_1, n_2, \ldots, n_H$.

LLS represents posterior estimation, in which p_h is estimated by p_h^*, where

$$p_h^* = E[p_h|n_1, \ldots, n_H] = \frac{n_h + \lambda}{N + \lambda H}, \tag{12}$$

where λ is a predefined non-negative parameter. In statistics, a widely used value for λ is 0.5 (the value 0.50922 is suggested in Krichevskiy 1998).

LLS is a generalization of Laplace's Law of Succession (the latter fixes λ to 1). LLS is more widely used in the statistical literature (Ristad 1995), and also applied in Machine Learning, such as in Naive Bayes Classifiers (Kohavi et al. 1997, and note that in the literature, Lidstone's Law is sometimes referred to as Laplace's Law). However, most researchers simply compute the expected value according to (12). In Theorem 2, we will show that the expected value (1-norm) is not sufficient to correctly evaluate a tree's accuracy on unseen data (the theorem shows that any effective split decreases the 1-norm error rate; consequently any effective split will not be eliminated if the 1-norm error rate is applied). In Sect. 4.3, we will also give an example where an end-cut improves the expected accuracy; this example shows that by considering both the expected accuracy and the average standard deviation of the accuracy we are able to identify redundant splits.

In Lidstone's Estimation, it is assumed (Good 1967, 1965) that the prior distribution of p_h is the Dirichlet distribution, denoted by

$$(p_1, \ldots, p_H) \sim Dir(\lambda, \lambda, \ldots, \lambda), \tag{13}$$

or

$$f(p_1, \ldots, p_H) = \alpha\delta\left(1 - \sum_{h=1}^{H} p_h\right) \prod_{h=1}^{H} p_h^{\lambda-1} I[p_h \geq 0], \tag{14}$$

where α is a constant so that the integral of $f(p_1, \ldots, p_H)$ is unity, and $\delta(x)$ is Dirac's delta function (or unit impulse function; Zemanian 1987) such that

$$\int_{-\infty}^{\infty} g(x)\delta(x)dx = g(0), \tag{15}$$

for all continuous functions g. Dirac's delta function ensures that the p_h's sum up to one, for the events Φ_h's are collectively exhaustive.

Based on (14), it has been proved that the posterior probabilities also follow the Dirichlet distribution in Good (1965). In particular,

$$(p_1, \ldots, p_H|n_1, \ldots, n_H) \sim Dir(n_1 + \lambda, \ldots, n_H + \lambda). \tag{16}$$

One can also easily prove that

$$(p_h, 1 - p_h|n_1, \ldots, n_H) \sim Dir(n_h + \lambda, N - n_h + \lambda(H - 1)). \tag{17}$$

This result leads to the following (Good 1967, (20)):

$$E[p_h^k|n_1, \ldots, n_H] = \prod_{m=0}^{k-1} \frac{n_h + \lambda + m}{N + \lambda H + m}, \tag{18}$$

$$E[(1 - p_h)^k|n_1, \ldots, n_H] = \prod_{m=0}^{k-1} \frac{N - n_h + \lambda(H - 1) + m}{N + \lambda H + m}. \tag{19}$$

The standard deviation of p_h can be computed as

$$\sigma[p_h|n_1,\ldots,n_H]=\sqrt{E[p_h^2|n_1,\ldots,n_H]-E[p_h|n_1,\ldots,n_H]^2}. \tag{20}$$

Equation (19) is the basis of our proposed error rate estimation and the corresponding k-norm pruning algorithm.

Note that $(p_1,\ldots,p_H)$ are assumed symmetric here, according to their equal parameters in the prior joint PDF in (14). In tree classification, we are consistent if we treat the events "$\mathbf{X}$ has class label j" as symmetric, which means that we have no bias towards any class before seeing any training examples. In contrast, we are inconsistent if we treat the events "$\mathbf{X}$ is correctly classified" versus "$\mathbf{X}$ is misclassified" as symmetric, for these events depend on the leaf predictions.

4 Error rate estimation—k-norm pruning algorithm

As explained in Sect. 2.3, we apply LLS to estimate the error rates. We also rely on the assumptions (Partitioning, Determinancy and Piece-Wise Independence) discussed in Sect. 3.1.1.

4.1 Terminology

We now estimate the misclassification rate $P[Err|\mathbf{X}]$ of a tree, by treating $P[Err|\mathbf{X}]$ as a random variable. In the rest of this section we use expected values extensively, and thus it is worth mentioning first that the expected value here is the integral across two random factors: the random input attribute vector $\mathbf{X}$ and all the unknown probabilities $\mathbf{P}$ (class probabilities $P[C_j|A_t]$, child probabilities $P[A_c|A_d]$, etc.). That is, for any random variable Q,

$$E_{\mathbf{X}}[Q]=\int Qf(\mathbf{X})d\mathbf{X}, \tag{21}$$

$$E_{\mathbf{P}}[Q]=\int Qf(\mathbf{P})d\mathbf{P}, \tag{22}$$

$$E[Q]=E_{\mathbf{X}}[E_{\mathbf{P}}[Q]]=E_{\mathbf{P}}[E_{\mathbf{X}}[Q]], \tag{23}$$

where $f(\mathbf{P})$ is actually a posterior PDF (given the observations with the training examples; we omit this condition for simplicity, because it is common among all equations in the rest of this paper). Similarly, the standard deviations are defined as follows:

$$\sigma_{\mathbf{P}}[Q]=\sqrt{E_{\mathbf{P}}[Q^2]-E_{\mathbf{P}}[Q]^2}, \tag{24}$$

$$\sigma[Q]=\sqrt{E[Q^2]-E[Q]^2}. \tag{25}$$

When conditions are involved, we put the conditions into the subscript. For example,

$$E_{\mathbf{X}|A_t}[Q]=\int Qf(\mathbf{X}|A_t)d\mathbf{X}. \tag{26}$$

Note that $\mathbf{P}$ is composed of all probabilities, including $P[C_j|A_t]$ for any node t, and the expression "$P[C_j|A_t]|A_t$" is not meaningful. Therefore, we treat $\mathbf{P}$ as independent of A_t and compute the expected value of a random variable within a node t as

$$E^t[Q]=E_{\mathbf{P}}[E_{\mathbf{X}|A_t}[Q]], \tag{27}$$

where the superscript t represents the condition "$|A_t$". When Q has the same subscript t, we omit the superscript t in $E^t[Q]$. For example, the expected error rate of the sub-tree rooted at t is represented by $E[r_t]$ rather than $E^t[r_t]$, because this expected value must, of course, be computed with the assumption that A_t is true.

4.2 Partitioning theorem

Theorem 1 *For any decision node d and any natural number k, under the Partitioning Assumption and the Determinacy Assumption, the following expression is valid*:

$$E_{\mathbf{X}|A_d}[r_d^k] = \sum_{c \in Children(d)} E_{\mathbf{X}|A_c}[r_c^k] P[A_c|A_d], \tag{28}$$

where r_t is the error rate of node t, defined as follows:

$$r_t = P[Err|A_t, \mathbf{X}]. \tag{29}$$

Proof The proof of Theorem 1 is provided in Appendix B. □

Note that $E_{\mathbf{X}|A_d}[Q]$ represents only the expectation across $\mathbf{X}$, and thus both sides in (28) are still random variables.

Corollary 1 *Under the Partitioning Assumption and the Determinacy Assumption, as well as the assumption that r_c and $P[A_c|A_d]$ are independent random variables for each child node c of the decision node d, the following expression is valid*:

$$E[r_d^k] = \sum_{c \in Children(d)} E[r_c^k] E_{\mathbf{P}}[P[A_c|A_d]]. \tag{30}$$

This corollary can be proved by computing the expected values with respect to $\mathbf{P}$ of both sides of (28), according to (27). Apparently, the error rate of the tree is equal to r_{root}. This corollary indicates that the error rate of a tree (or a sub-tree) can be estimated recursively. We apply LLS to obtain explicit expressions for error rate estimation, as follows.

For a leaf l, under the Piece-wise Independence Assumption (6),

$$E[r_l^k] = E[(1 - P[C_{Label(l)}|A_l, \mathbf{X}])^k] \approx E[(1 - P[C_{Label(l)}|A_l])^k]. \tag{31}$$

Recall that all expected values here are conditional, based on the training examples. It is well known that to estimate the probabilities of a set of mutually exclusive and collectively exhaustive events with N independent trials, the numbers of occurrences of the events are sufficient. Therefore, the condition "given training examples" can be replaced with "given $n_{j,l}$". According to (19), by defining $H = J$ and $p_j = P[C_j|A_l]$, $j = 1, 2, \ldots, J$, we have

$$\begin{aligned} E[r_l^k] &\approx E[(1 - P[C_{Label(l)}|A_l])^k] \\ &= \prod_{i=0}^{k-1} \frac{n_{Label(l),l} + \lambda(J-1) + i}{n_l + \lambda J + i} \\ &= \prod_{i=0}^{k-1} \frac{n_l - \max_j n_{j,l} + \lambda(J-1) + i}{n_l + \lambda J + i}. \end{aligned} \tag{32}$$

For a decision node d, since $\mathbf{P}$ includes $P[A_c|A_d]$, the other probabilities in $\mathbf{P}$ do not contribute to $E_{\mathbf{P}}[P[A_c|A_d]]$ (because $E_Y[E_Z[Z]] = E_Z[Z]$ for any two random variables Y and Z even if they are dependent on each other). Applying LLS again, we get

$$E_{\mathbf{P}}[P[A_c|A_d]] = \frac{n_c + \eta}{n_d + \eta K_d}, \tag{33}$$

where K_d is the number of immediate children of decision node d, n_t is the number of training examples covered by node t, and η has an analogous functionality as λ (we use a different notation because λ has been used to denote the parameter in (32), and here we allow η to take a different value). For simplicity, from now on, we use $P^*[A_c|A_d]$ to denote $E_{\mathbf{P}}[P[A_c|A_d]]$.

Using Corollary 1, the standard deviation of the error rate can be computed as follows:

$$E[r_d] = \sum_{c \in Children(d)} E[r_c] P^*[A_c|A_d], \tag{34}$$

$$E[r_d^2] = \sum_{c \in Children(d)} E[r_c^2] P^*[A_c|A_d], \tag{35}$$

$$\sigma[r_d] = \sqrt{E[r_d^2] - E[r_d]^2}. \tag{36}$$

It is important to note that generally,

$$\sigma[r_d] \neq \sum_{c \in Children(d)} \sigma[r_c] P^*[A_c|A_d]. \tag{37}$$

Please see Sect. 4.3 for a detailed application of the above equations.

4.3 An example

Consider now a more specific example of a 2-class classification problem for which a binary tree is built. Suppose a decision node d of the tree receives 98 training examples of class 1 and 1 example of class 2, and its split separates the two classes. Assume $\lambda = \eta = 0.5$. According to (6), before splitting,

$$\begin{aligned} E[r_d] &= E_{\mathbf{P}}[E_{\mathbf{X}}[P[Err|A_d, \mathbf{X}]]] = E_{\mathbf{P}}[P[Err|A_d]] \\ &= E_{\mathbf{P}}[1 - P[C_1|A_d]] = \frac{1 + 0.5}{99 + 0.5 \times 2} = 0.015000, \end{aligned} \tag{38}$$

$$\begin{aligned} E[r_d^2] &= E_{\mathbf{P}}[E_{\mathbf{X}}[P[Err|A_d, \mathbf{X}]^2]] \approx E_{\mathbf{P}}[P[Err|A_d]^2] \\ &= E_{\mathbf{P}}[(1 - P[C_1|A_d])^2] = \frac{(1 + 0.5)(1 + 0.5 + 1)}{(99 + 0.5 \times 2)(99 + 0.5 \times 2 + 1)} \\ &= 0.00037129, \end{aligned} \tag{39}$$

$$\sigma[r_d] = \sqrt{E[r_d^2] - E[r_d]^2} \approx 0.012095. \tag{40}$$

After splitting, for the left child c_1,

$$E[r_{c_1}] = \frac{0 + 0.5}{98 + 0.5 \times 2} = 0.0050505, \tag{41}$$

$$E[r_{c_1}^2] \approx \frac{(0+0.5)(0+0.5+1)}{(98+0.5\times 2)(98+0.5\times 2+1)} = 0.00075758, \tag{42}$$

$$P^*[A_{c_1}|A_d] = \frac{98+0.5}{99+0.5\times 2} = 0.98500. \tag{43}$$

After splitting, for the right child c_2,

$$E[r_{c_2}] = \frac{0+0.5}{1+0.5\times 2} = 0.25000, \tag{44}$$

$$E[r_{c_2}^2] \approx \frac{(0+0.5)(0+0.5+1)}{(1+0.5\times 2)(1+0.5\times 2+1)} = 0.12500, \tag{45}$$

$$P^*[A_{c_2}|A_d] = \frac{1+0.5}{99+0.5\times 2} = 0.015000. \tag{46}$$

For the sub-tree,

$$E[r_d] = 0.0050505\times 0.985 + 0.25\times 0.015 = 0.0087247, \tag{47}$$

$$E[r_d^2] \approx 0.00075758\times 0.985 + 0.125\times 0.015 = 0.0019496, \tag{48}$$

$$\sigma[r_d] \approx \sqrt{0.0019496 - (0.0087247)^2} = 0.04328. \tag{49}$$

Although the expected value of the error rate decreases by 0.006275 after the split, the average standard deviation increases by 0.03119, which is almost 5 times larger than the decreased amount in the expected value. Thus, while the expectation of the error is lower, there is a relatively high probability that the true error has increased, rather than decreased. There is little evidence that this split will improve the generalization.

4.4 k-norm estimation in tree pruning—k-norm pruning algorithm

When enough (thousands of) samples are given, one does not systematically expect higher values than the expected value. In a tree, however, no matter how many samples are initially given, after splitting there might be tree leaves that contain only a few samples. In this case, the expected values calculated at these leaves are not reliable, as shown in the previous example. Therefore, it is reasonable to consider combining the expected value and the standard deviation to estimate the error rate. For example, as shown in Sect. 4.3, $E[r^2] = E[r]^2 + \sigma[r]^2$ can be used to identify redundant splits, and its square root (2-norm) can serve as our error rate estimate. In general, k-norm estimation in tree pruning, on which the k-norm pruning algorithm is based, estimates $\|r\|_k = \sqrt[k]{E[|r|^k]}$ (which is equal to $\sqrt[k]{E[r^k]}$ because in this paper $r \geq 0$). Some observations, pertinent to $\sqrt[k]{E[r^k]}$ value, are listed below:

1. When k is fixed, minimizing the k-norm error $\|r\|_k$ is equivalent to minimizing the k-th moment $E[r^k]$. According to Theorem 1, the moment can be computed recursively, by starting at the leaves of the tree and moving upwards towards the root of the tree; the error rate of the sub-trees under any decision node can be summed up independently. To optimize any sub-tree rooted at d, we can first optimize the sub-trees under d and then optimize the node d (by deciding whether to replace it with a leaf), which guarantees global optimality. This property is very important for our k-norm pruning algorithm.
2. When $k = 2$, $\|r\|_2 = \sqrt{E[r]^2 + \sigma[r]^2}$ takes both the expected value and the standard deviation into account. When $k > 2$, other statistical information such as skewness and/or kurtosis is also included.

input : a tree rooted at node t
output: the optimal pruned tree (modified from the input),
and its k-th moment error rate (returned value of this function)

Compute R_{leaf} according to (32);
if *t is a decision node* **then**
  Compute $R_{tree} = \sum_{c \in Children(t)} E_{\mathbf{P}}[P[A_c|A_d]]$`PruneTree`$(c)$ according to (33);
  if $R_{tree} < R_{leaf} - \varepsilon$ *(or* $\sqrt[k]{R_{tree}} < \sqrt[k]{R_{leaf}} - \varepsilon$*)* **then**
    return R_{tree};
  end
  Replace the subtree rooted at t with a leaf;
end
return R_{leaf};

Algorithm 1: PruneTree

By using the k-norm error rate estimation, in order to get the optimally pruned tree at t, we can first get the optimally pruned trees below t, and then consider whether pruning t yields a lower k-norm error rate. The algorithm is shown in Algorithm 1.

In the rest of this paper, we choose $k = 2$ for the k-norm estimation, since it takes into account both the expected value as well as the standard deviation of the error rate. We do not use higher than $k = 2$ values for the error rate estimation, because in this paper, we do not see the benefits of taking the skewness into account. Moreover, there is a potential risk for higher values of k: for any random variable Q, the estimation of $\|Q\|_k$, using a finite set of examples, becomes more sensitive to noise and computational errors, which can also be seen in the approximation in (32). We do not use $k = 1$ for the k-norm estimation because it is too optimistic as an error rate estimate, which is empirically verified in Esposito et al. (1997), and theoretically demonstrated by the following theorem.

Theorem 2 *Under the Partitioning Assumption and the Determinacy Assumption, if* $0 \leq \eta \leq \lambda J$, $\lambda \leq 1/(K_t - 1)(J - 1)$ *at a node* t, *then any effective tree split decreases the* 1*-norm error rate. A tree split is called effective if it reduces the training misclassifications.*

Proof The proof can be found in Appendix C. □

Since MEP can be represented by the 1-norm pruning with $\eta = 0$ and flexible λ, we see that it never prunes any end-cuts (one that separates only a few training examples from the rest), as long as they are effective. See Sect. 4.3 for example.

4.5 Application in tree prediction

Given an input $\mathbf{X}$, we can predict not only the class label but also an estimated error rate for this prediction. Here $\mathbf{X}$ is deterministic and therefore we only compute $E_{\mathbf{P}}[r^k]$ where $r = P[Err|\mathbf{X}]$. We also apply the k-norm estimation here. Under the assumptions of Sect. 3.1.1, only one leaf $l_{\mathbf{X}}$ receives $\mathbf{X}$. Therefore,

$$E_{\mathbf{P}}[r^k] = E_{\mathbf{P}}[(1 - P[C_{Label^*(\mathbf{X})}|\mathbf{X}, A_{l_{\mathbf{X}}}])^k]$$
$$\approx E_{\mathbf{P}}[(1 - P[C_{Label(l_{\mathbf{X}})}|A_{l_{\mathbf{X}}}])^k]$$

$$= \prod_{i=0}^{k-1} \frac{n_{l_{\mathbf{X}}} - \max_j n_{j,l_{\mathbf{X}}} + \lambda(J-1) + i}{n_{l_{\mathbf{X}}} + \lambda J + i}. \tag{50}$$

In practice, there are two reasonable ways to output the estimated error rate: 1) Output the 2-norm error rate $\|r\|_2 = \sqrt{E_{\mathbf{P}}[r^2]}$ only, or 2) Output $E_{\mathbf{P}}[r]$ and $\sigma_{\mathbf{P}[r]} = \sqrt{E_{\mathbf{P}}[r^2] - E_{\mathbf{P}}[r]^2}$, the latter (average standard deviation) representing the reliability or a range in the form of 0.6 ± 0.1; the smaller the range is, the more confident we are when using the expected value to estimate the generalization error rate, meaning that the expected value is more reliable.

5 Experiments

In this section, we compare our proposed 2-norm pruning algorithm with two other classical pruning algorithms: CCP of CART and EBP of C4.5.

5.1 Test procedure

We compare the algorithms with respect to three metrics: the elapsed time in pruning (computational complexity of pruning), the accuracy of the pruned tree, and the resulting tree size. Reducing the tree size is not our goal, and it is shown here only as a companion measure to the accuracy comparison.

In our experiments, each database is separated into a training set and a test set. To reduce the randomness of the partitioning of the data in a training and test set, we repeated the experiments 20 times with different partitioning each time. To ensure that the classes in the training set have approximately the same distribution as in the test set, we shuffle the examples before partitioning.

Apparently, if the random partitioning is simply repeated, some of the examples might appear more often in the training set than in the test set or the opposite. Although V-fold cross-validation can address this problem, we cannot control the training ratio (the proportion of the training set) when V is fixed, and this ratio is large when V is high. For example, when $V = 10$, the training set is 9 times larger than the test set, with 90% training ratio. Therefore, we slightly digress from the cross-validation approach, as explained in the following: We partition each database only once into 20 subsets with approximately equal size. In the i-th ($i = 0, 2, \ldots, 19$) test, we use subsets $i \bmod 20$, $(i+1) \bmod 20, \ldots, (i+m-1) \bmod 20$ for training and the others for testing, where m is predefined (e.g., if the training ratio is 5%, $m = 1$; if the training ratio is 50%, $m = 10$). As illustrated in Fig. 3, this approach guarantees that each example appears exactly m times in the training set and exactly $20 - m$ times in the test set. Furthermore, it allows the training ratio to vary. The procedure is delineated in Algorithm 2.

Note that a training ratio of 5% is a challenging setting (very few training cases) for which the robustness of the algorithms can be examined. Furthermore increasing the training set size from 5% to 50% (a 10-fold increase of the training cases) allows us to investigate how well the pruning algorithms scale with a 10-fold increase of training data size.

To make a fair and justified comparison on speed, we have implemented all the algorithms in C++ as a Matlab MEX file, and we ran all experiments on the same computer (Intel Pentium 4 at 2.59 GHz, 1.5 GB Memory, and Windows XP). No file I/O, screen display, or keyboard input, but only number crunching is timed. We used a high resolution timer whose result is accurate to one microsecond.

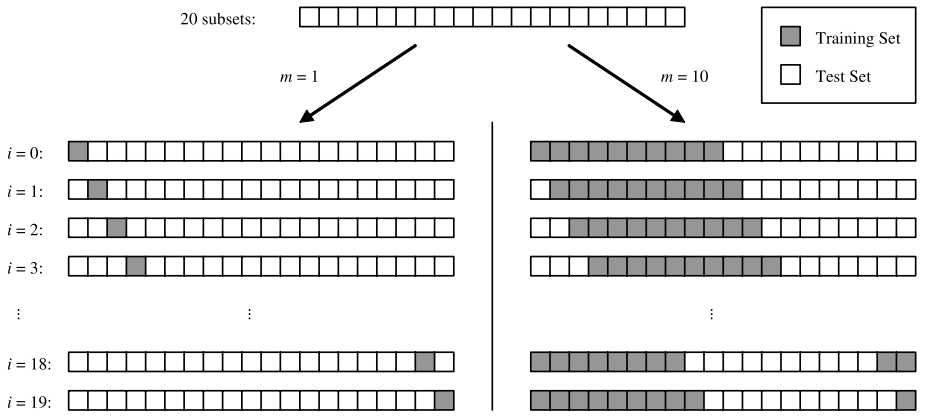


Fig. 3 Visualization of the followed procedure to partition the databases into training and test sets

```
foreach Database D do
    Shuffle the examples in D;
    Partition D into 20 subsets D_0, D_1, ..., D_19;
    for m = 1, 10 do
        for i = 0 to 19 do
            Set TrainingSet = ⋃_{j=0}^{m-1} D_{(i+j) mod 20};
            Set TestSet = D − TrainingSet;
            Grow a full tree with CART and TrainingSet;
            Use CCP/EBP/2-norm for pruning the tree;
            Evaluate each pruned tree with TestSet;
        end
    end
end
```

Algorithm 2: Experimental procedure to create training and test subsets from a database set

5.2 Databases

To ensure that our experimental results are statistically significant, we require that the databases must have at least 2000 examples. Furthermore, according to the *Determinacy Assumption*, our algorithm does not handle missing values at this time, and therefore we choose only the databases without missing values. The statistics of the databases that we experimented with are listed in Table 1.

The databases g#c## are Gaussian artificial databases. The acronym g2c15 means 2 classes and 15% minimum error rate (defined as the error rate of the Bayes optimal classifier). These databases have two attributes x and y. In each class, x and y are independent Gaussian random variables, whose means vary among the classes. The other databases are benchmark databases obtained from the UCI Repository (Newman et al. 1998), according to our criteria mentioned in the beginning of this subsection.

Table 1 Statistics of the Databases used in our experiments

Database	Examples	Numerical Attributes	Categorical Attributes	Classes	Majority Class %
abalone	4177	7	1	3	34.6
g2c15	5000	2	0	2	50.0
g2c25	5000	2	0	2	50.0
g6c15	5004	2	0	6	16.7
g6c25	5004	2	0	6	16.7
kr-vs-kp	3196	0	36	2	52.2
letter	20000	16	0	26	4.1
optdigits	5620	64	0	10	10.2
pendigits	10992	16	0	10	10.4
satellite	6435	36	0	6	23.8
segment	2310	19	0	7	14.3
shuttle	58000	9	0	7	78.6
splice	3190	0	60	3	51.9
waveform	5000	21	0	3	33.9

5.3 Parameters

For the 2-norm algorithm, the η value is not critical and is set to 0.5. However, the parameter λ affects the error rate estimate. When $\lambda = 0$, the estimated error rate is approximately the training error rate, which results in under-pruning. When $\lambda = \infty$, the estimated error rate is $(J - 1)/J$ for all nodes and thus the tree will be pruned to only one node.

In our experiments, we set $\lambda = \lambda_0 V/J$, where λ_0 is a constant and V is a heuristic indicating how much the classes overlap with each other. We decrease λ for the databases with more classes to prevent over-pruning, because given the same λ, the same training error rate and the same training data size, the estimated error rate increases with respect to J as seen in (32). For highly-overlapping databases, we increase λ to prune more nodes. The heuristic V is computed as $L/(JN)$, where L is the number of leaves in the fully grown tree. Apparently, L increases with J and with N. When J and N are fixed, L is greater for the databases with higher overlap among their classes. Our preliminary experiments based on the Iris database show that λ_0 can be set to 100, so that λ is approximately 0.5 for this database. The details of the experiments on the Iris database are shown in Appendix A. Finally, for CCP and EBP, the default parameters were used.

5.4 Results

5.4.1 Accuracy

The accuracy of the trees pruned by each algorithm is listed in Table 2. We show the mean, the standard deviation and the t-test result of the 20 runs. We choose to call the difference between two accuracies significant if the difference is above 1% (practically significant) and the p-value of the t-test turns out to be at most 5% (statistically significant).

It can be seen that when the training ratio is small (5%), the 2-norm pruning algorithm outperforms CCP 5 times out of the 14 database experiments, while CCP never outperforms the 2-norm algorithm. The cause is that when the size of the training set is very small,

Table 2 Comparison of Tree Accuracy for CCP, 2-norm and EBP

R	Database	CCP		2-norm		EBP		2-norm : CCP			2-norm : EBP		
		Mean	Std	Mean	Std	Mean	Std	Diff	P	±	Diff	P	±
5	abalone	59.1	1.5	59.5	2.0	56.6	1.5	0.4	48.4		2.9	0.0	+
5	g2c15	83.4	1.1	84.7	0.9	82.5	1.1	1.3	0.0	+	2.2	0.0	+
5	g2c25	72.5	1.4	74.0	0.9	69.4	2.4	1.5	0.0	+	4.6	0.0	+
5	g6c15	79.7	1.3	79.2	1.4	78.8	1.7	−0.5	28.1		0.4	35.6	
5	g6c25	69.4	1.1	69.4	1.4	68.1	1.2	0.0	97.5		1.3	0.2	+
5	kr-vs-kp	93.4	1.6	93.5	1.5	93.9	1.4	0.1	80.9		−0.4	43.0	
5	letter	62.8	1.2	62.8	0.9	63.3	1.0	−0.1	81.4		−0.5	10.9	
5	optdigits	70.2	1.7	71.7	1.9	72.0	2.1	1.5	1.3	+	−0.3	58.9	
5	pendigits	83.8	1.5	83.2	1.4	84.7	1.2	−0.6	23.0		−1.5	0.1	−
5	satellite	77.9	1.4	79.2	1.3	78.4	1.6	1.3	0.3	+	0.8	8.1	
5	segment	85.5	2.6	86.2	2.4	86.8	2.4	0.7	38.8		−0.6	46.2	
5	shuttle	99.7	0.1	99.6	0.1	99.7	0.1	−0.1	2.9		−0.1	0.0	
5	splice	82.6	3.7	83.1	2.9	80.5	3.4	0.5	64.1		2.5	1.5	+
5	waveform	69.0	1.7	70.7	1.2	70.0	1.5	1.7	0.1	+	0.8	8.1	
5		Summary						5 wins, 0 losses			5 wins, 1 loss		
50	abalone	63.0	1.0	62.6	0.6	58.3	1.4	−0.4	19.2		4.3	0.0	+
50	g2c15	84.8	0.7	85.4	0.4	82.6	0.7	0.7	0.0		2.8	0.0	+
50	g2c25	74.4	0.8	74.4	0.7	71.6	1.0	0.0	99.3		2.8	0.0	+
50	g6c15	82.4	0.5	82.2	0.6	81.1	0.8	−0.1	50.9		1.1	0.0	+
50	g6c25	72.7	0.9	73.0	0.6	70.5	0.5	0.2	30.4		2.5	0.0	+
50	kr-vs-kp	99.1	0.2	98.5	0.6	99.2	0.2	−0.6	0.1		−0.7	0.0	
50	letter	84.1	0.4	83.5	0.3	84.0	0.4	−0.6	0.0		−0.6	0.0	
50	optdigits	88.1	1.0	87.9	0.8	88.3	0.8	−0.2	49.6		−0.4	16.9	
50	pendigits	95.1	0.3	94.7	0.4	95.1	0.3	−0.4	0.1		−0.4	0.0	
50	satellite	85.3	0.6	85.6	0.6	85.0	0.6	0.3	7.9		0.7	0.2	
50	segment	94.0	1.0	94.2	0.7	94.6	0.6	0.1	60.9		−0.4	4.5	
50	shuttle	99.9	0.0	99.9	0.0	99.9	0.0	0.0	7.8		0.0	9.2	
50	splice	94.3	0.4	94.1	0.4	92.9	0.7	−0.2	15.1		1.2	0.0	+
50	waveform	75.8	1.0	76.5	0.6	75.0	0.6	0.8	0.5		1.6	0.0	+
50		Summary						0 wins, 0 losses			7 wins, 0 losses		

• R is the training ratio (%)
• Mean and Std are the mean and the standard deviation, respectively, of the accuracy (%) over the 20 runs in each database
• Diff is the difference of the mean accuracy between the 2-norm pruning algorithm and the one being compared to
• P is the p-value (%) of the t-test of the 20 observations. It is the probability of the observations given that the null hypothesis is true. Small values of P cast doubt on the validity of the null hypothesis
• ± is the comparison result. A comparison result is shown only when significant, that is, when |Diff| is at least one (indicating practical significance) and P is at most 5% (indicating statistical significance). If the result is significant and Diff is positive, "+" is shown (which means 2-norm wins); if the result is significant and Diff is negative, "−" is shown

Table 3 Comparison of Tree Size for CCP, 2-norm and EBP

R	Database	CCP		2-norm		EBP		2-norm : CCP			2-norm : EBP		
		Mean	Std	Mean	Std	Mean	Std	Diff	P	±	Diff	P	±
5	abalone	6.9	5.2	9.5	3.2	43.2	5.8	2.6	6.3	−	−33.7	0.0	+
5	g2c15	7.0	2.6	2.2	0.7	14.2	5.2	−4.9	0.0	+	−12.0	0.0	+
5	g2c25	6.3	3.6	2.7	1.9	28.5	7.4	−3.6	0.0	+	−25.8	0.0	+
5	g6c15	8.8	1.9	7.4	1.4	20.9	4.3	−1.4	1.2	+	−13.5	0.0	+
5	g6c25	8.5	2.5	8.6	1.9	33.2	5.3	0.1	88.8		−24.6	0.0	+
5	kr-vs-kp	7.2	2.4	6.0	1.6	9.3	2.0	−1.2	6.8		−3.4	0.0	+
5	letter	171.9	81.9	232.2	11.0	279.4	8.6	60.3	0.2	−	−47.2	0.0	+
5	optdigits	19.1	4.9	31.1	5.5	47.7	4.4	12.0	0.0	−	−16.6	0.0	+
5	pendigits	39.8	13.3	36.6	3.5	55.6	4.7	−3.2	30.5		−19.0	0.0	+
5	satellite	9.0	4.0	17.5	3.3	34.6	4.0	8.5	0.0	−	−17.1	0.0	+
5	segment	8.5	1.8	9.4	1.3	12.6	1.3	0.9	9.5		−3.3	0.0	+
5	shuttle	8.6	2.4	6.3	1.7	11.7	1.5	−2.3	0.2	+	−5.4	0.0	+
5	splice	8.6	4.2	7.9	1.2	16.7	1.8	−0.7	48.1		−8.8	0.0	+
5	waveform	7.5	4.1	13.0	2.2	31.2	1.9	5.5	0.0	−	−18.2	0.0	+
5		Summary						4 wins, 5 losses			14 wins, 0 losses		
50	abalone	12.2	7.3	72.1	11.2	382.8	13.8	59.9	0.0	−	−310.7	0.0	+
50	g2c15	10.6	7.2	2.0	0.0	114.0	15.3	−8.6	0.0	+	−112.0	0.0	+
50	g2c25	13.0	6.4	25.0	17.7	230.2	26.6	12.1	0.7	−	−205.2	0.0	+
50	g6c15	16.1	3.0	22.2	6.2	145.5	13.4	6.1	0.0	−	−123.4	0.0	+
50	g6c25	13.0	4.2	32.2	8.6	256.0	19.6	19.2	0.0	−	−223.8	0.0	+
50	kr-vs-kp	27.0	3.4	21.4	4.5	27.9	2.7	−5.7	0.0	+	−6.6	0.0	+
50	letter	1283.8	28.4	1083.0	26.7	1294.1	25.9	−200.8	0.0	+	−211.1	0.0	+
50	optdigits	94.3	16.1	128.4	7.7	210.9	11.0	34.1	0.0	−	−82.5	0.0	+
50	pendigits	177.3	38.9	141.6	9.9	199.0	7.2	−35.7	0.0	+	−57.4	0.0	+
50	satellite	44.8	17.6	96.3	7.4	221.5	10.3	51.5	0.0	−	−125.2	0.0	+
50	segment	28.3	9.6	29.1	2.4	43.2	3.1	0.9	70.3		−14.1	0.0	+
50	shuttle	23.3	4.5	19.6	1.8	21.2	2.6	−3.7	0.1	+	−1.7	2.7	+
50	splice	15.7	3.2	27.4	2.5	52.5	4.2	11.7	0.0	−	−25.1	0.0	+
50	waveform	28.3	11.3	82.3	7.1	234.9	6.6	54.1	0.0	−	−152.6	0.0	+
50		Summary						5 wins, 8 losses			14 wins, 0 losses		

• R is the training ratio (%)
Mean and Std are the mean and the standard deviation, respectively, of the tree size (number of leaves) over the 20 runs in each database
• Diff is the difference of the mean size between the 2-norm pruning algorithm and the one being compared to
• P is the p-value (%) of the t-test of the 20 observations. It is the probability of the observations given that the null hypothesis is true. Small values of P cast doubt on the validity of the null hypothesis
• ± is the comparison result. A comparison result is shown only when significant, that is, when |Diff| is at least one and P is at most 5%. If the result is significant and Diff is negative, "+" is shown (which means 2-norm wins); if the result is significant and Diff is positive, "−" is shown

cross-validation is not quite reliable in evaluating the candidate pruned trees, because each validation set has only 1/10 of the training data. We examined the error rate estimation of cross-validation and our algorithm and it turns out that ours tends to be better. For example,

Table 4 Comparison of Elapsed Time for CCP, 2-norm and EBP

Database	CCP			2-norm			EBP		
	T_1	T_2	Inc	T_1	T_2	Inc	T_1	T_2	Inc
abalone	8.5E−2	1.5E+0	17.5	1.3E−4	7.6E−4	6.0	1.3E−3	3.0E−2	22.8
g2c15	2.6E−2	5.9E−1	22.5	1.1E−4	5.7E−4	5.4	7.7E−4	1.8E−2	22.7
g2c25	3.6E−2	8.1E−1	22.3	1.3E−4	8.6E−4	6.5	1.4E−3	3.9E−2	27.1
g6c15	3.3E−2	6.2E−1	19.1	1.1E−4	6.9E−4	6.1	8.5E−4	1.9E−2	22.1
g6c25	4.0E−2	7.8E−1	19.3	1.4E−4	1.0E−3	7.2	1.3E−3	3.2E−2	25.4
kr-vs-kp	3.6E−2	3.6E−1	10.1	7.3E−5	1.0E−4	1.4	8.1E−4	8.4E−3	10.3
letter	1.2E+0	1.5E+1	12.2	5.1E−4	2.8E−3	5.4	1.6E−2	2.5E−1	15.6
optdigits	5.7E−1	8.7E+0	15.3	1.2E−4	3.4E−4	2.9	1.4E−3	2.6E−2	18.3
pendigits	4.9E−1	6.8E+0	13.8	1.3E−4	3.7E−4	2.7	2.2E−3	3.8E−2	17.0
satellite	5.5E−1	9.3E+0	16.9	2.8E−4	4.2E−4	1.5	1.2E−3	2.9E−2	23.5
segment	8.2E−2	1.2E+0	14.4	6.3E−5	1.4E−4	2.2	3.6E−4	4.2E−3	11.7
shuttle	6.7E−1	2.0E+1	29.5	6.1E−5	8.9E−5	1.4	3.2E−3	1.0E−1	32.4
splice	9.8E−2	1.0E+0	10.2	7.9E−5	1.8E−4	2.3	1.3E−3	1.3E−2	10.0
waveform	2.6E−1	5.7E+0	21.7	9.3E−5	4.0E−4	4.3	9.5E−4	2.4E−2	25.4

- T_1 and T_2 are the mean of the elapsed time (seconds) in the 20 runs of the pruning algorithms for training ratio = 5% and 50%, respectively
- Inc equals T_2/T_1
- All numbers are rounded to one digit after the decimal

on the database *g2c15*, the root-mean-square-error (across the 20 runs) between the estimated CART error rate and the actual CART error rate is 4.45% for cross-validation and is 2.15% for 2-norm estimation. On the other hand, when the training ratio is large (50%), the 2-norm pruning algorithm has similar accuracy as CCP (neither one of these algorithms outperforms the other for any database experiment).

The 2-norm pruning algorithm also outperforms EBP in accuracy regardless of the training ratio used (5 times when the training ratio is 5% and 7 times when the training ratio is 50%), primarily because EBP tends to under-prune (see Table 3).

5.4.2 Tree size

Although our primary goal is *not* to minimize the tree size but the error rate, we show the size of the pruned trees in Sect. 3. The 2-norm pruning algorithm considers the estimated error rate only, and thus the resulting size is sometimes larger than that of CART. We also see (from Table 3) that C4.5 always generates larger trees than the other two.

5.4.3 Speed

The mean elapsed time (in seconds) of the three pruning algorithms is shown in Table 4. For each algorithm, the elapsed time is very stable in 20 runs, and thus we do not show the standard deviation that is negligible. For the same reason, and because the mean time is quite different among the three algorithms, all comparisons are practically significant and statistically significant.

Table 4 indicates that the 2-norm pruning algorithm is significantly faster than the other two (at least hundreds of times faster than CCP and usually tens of times faster than EBP).

The reason is that the 2-norm pruning algorithm does not need any training examples to be presented, and it finishes the tree optimization within one traversal of the tree. Although most of the times reported in Sect. 4 are small, our experiments show the apparent advantage of the 2-norm algorithm that is expected for huge databases and/or slow machines.

Our experiments also show that the 2-norm pruning algorithm scales better than CCP and EBP. The Inc Columns in Table 4 show that when the size of the training set is increased by ten times, the elapsed time of the 2-norm pruning algorithm is increased by less than ten times, while the time of the other two algorithms is increased by more than ten times. The explanation is as follows.

- Since the 2-norm pruning algorithm requires only one traversal of the fully grown tree and the visit of each node has a constant number of operations, the time complexity of the 2-norm pruning algorithm is $\Theta(M)$ where M is the number of nodes in the fully grown tree, and usually $M \ll N$, where N is the number of training examples (in the worst case, $M = \Theta(N)$).
- The 10-fold cross-validation in CCP involves 10 times of tree growing (Stone 1978; Breiman et al. 1984). When at least one numerical attribute is present, due to the sorting involved, the time complexity of the growing phase is $\Omega(N \log N)$, that is, $\Theta(N \log N)$ or higher; the worst case complexity $O(N^2 \log N)$ is justified in Duda et al. (2000), p. 406. When only categorical attributes are present, the time complexity is $\Omega(N)$. This explains why CCP's scalability is almost linear for the databases *kr-vs-kp* and *splice* (which have only categorical attributes) but worse for other databases. Note that although CART grows 10 trees and produces 10 corresponding pruning sequences, for each sequence CART does not have to evaluate all trees in the sequence with the corresponding pruning set, but only the fully grown tree in the sequence in order to compute the error rate for all its pruned trees. Therefore, the time of the evaluation step in CCP does not have a higher order than the growing step.
- EBP passes the training examples through the tree to evaluate the grafting in each decision node. Grafting in EBP considers the following three cases: a) the split test is used and the examples are dispatched to the subtrees; b) all examples are sent to the major child; c) the decision node is pruned to a leaf. The training examples must be visited to evaluate Option b) and thus the time complexity is $\Omega(N)$ (that is, it has a linear or higher order; the worst case complexity $O(N^2)$ can be derived similarly to the growing algorithm).

The speed comparison shows that the 2-norm pruning algorithm scales very well to large databases. Apparently, we can assume that one traversal of the tree is necessary to find the optimal pruned tree, that is, if a sub-tree is not visited, we do not have enough information to decide whether to prune it. Therefore, the 2-norm pruning algorithm has the minimal time complexity. No other algorithms can have a lower order of time complexity.

5.5 Additional experiments

5.5.1 Performance on large databases

To verify our previous statements about time complexity, we used 2 additional large databases, *g2c15* (artificial database) and *covtype* (UCI benchmark) to test the scalability of each algorithm. For each database, the number of training examples starts from 512 and doubles each time up to 524288 ($= 2^{19}$) examples. For *g2c15*, the number of test examples is always the same as that of training examples; for *covtype*, the examples not used for training are used for testing. This experiment is very time consuming, and thus is not repeated.

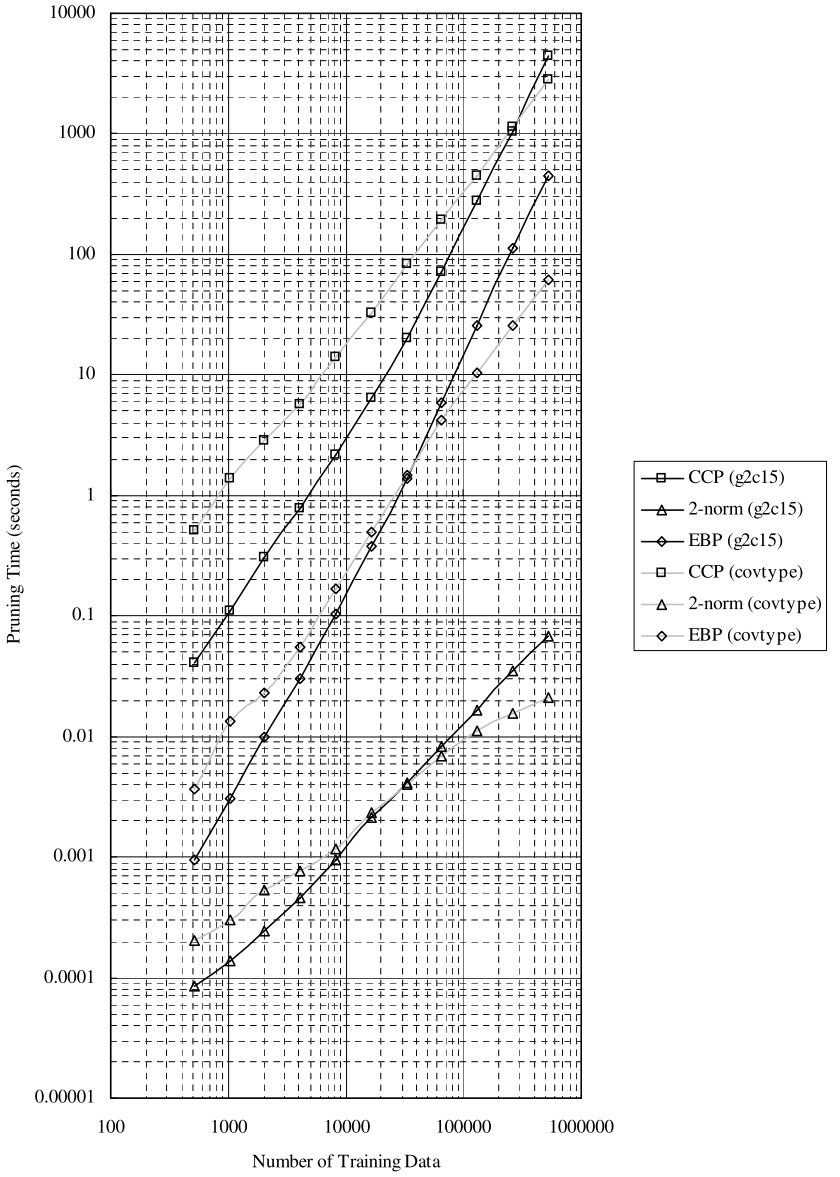


Fig. 4 Pruning Time of 2-norm, CCP and EBP on Large Databases (Gaussian database and Covertype database) as the training set size increases

The speed difference shown in Fig. 4 between our algorithm and CCP and EBP is apparent. The time required by our algorithm is linear in the size of these databases (because for these databases, the size of the fully grown tree turns out to be linear in the size of the database). At the same time, the time required by CCP and EBP increases non-linearly (note that both axes have logarithmic scales, and a straight line (i.e., $\log(y) = 2\log(x)$) may represent

a quadratic function in linear scale (i.e., $y = x^2$). Therefore, for these two large databases it is obvious that as the training set size becomes larger, the advantage of our algorithm is more pronounced. For instance, when the training set size of the *covtype* database is 1024 patterns the 2-norm requires 0.0003 seconds, versus 0.013 seconds and 1.38 second required by EBP and CCP, respectively. On the other hand for a training set size of 524,288 patterns and the *covtype* database, the 2-norm algorithm requires 0.021 seconds, compared to 61.3 seconds and 2848 seconds required by EBP and CCP, respectively.

As shown in Fig. 5, for the two large databases, the accuracy of 2-norm pruning algorithm is always similar to that of CCP, usually within 1%, and their tree sizes are also close. Note that for *g2c15*, 2-norm and CCP always produces the same tree with only 2 leaves, approaching the Bayes optimal classifier. EBP always produces the largest tree with no significantly better accuracy (2% worse on *g2c15*) than CCP and 2-norm.

5.5.2 Comparison with EBP featuring a varying CF value

It has been shown that EBP's accuracy can be improved by setting its *CF* parameter to very small values (Hall et al. 2003). Although this setting does not yield a meaningful error rate prediction, we tested these settings and compared to CCP and 2-norm pruning.

We used the real databases *waveform* and *letter*. We ran EBP, CCP, and our 2-norm pruning for 20 times with different partitioning, using the approach shown in Fig. 3 with $m = 10$. At each time, 50% of the data are used for training and the rest for testing. For 2-Norm and CCP, only one parameter setting is applied. For EBP, the *CF* parameter is varied as $100 \times 2^{-i}, i = 1, 2, \ldots, 20$ (including the default value 25). The average accuracy among the 20 runs is plotted against the *CF* value in Fig. 6.

Regarding EBP, the following observations can be made from Fig. 6:

1. The default value of *CF* does not necessarily maximize EBP's accuracy;
2. The EBP accuracy is not necessarily a decreasing function of *CF* (that is, we cannot set *CF* to extremely small values to obtain high accuracy);
3. Even when *CF* is optimized, EBP's accuracy is lower than or close to 2-Norm and CCP accuracy (for the databases shown in Fig. 6);
4. EBP's size tends to decrease when *CF* decreases; even when *CF* is very small EBP's size is still comparable to the size of 2-norm and CCP.

The time of EBP does not change significantly with respect to *CF*. Hence, the time comparisons depicted in Table 4 for EBP, CCP and 2-norm are still valid, independently of the *CF* value used.

5.5.3 Other databases

To compare our algorithm with other pruning algorithms (beyond CCP and EBP) we used the results included in Esposito et al. (1997). In particular, we ran the databases in Esposito et al. (1997) using the same protocol used there (70% of the data were used for training; experiments were repeated 25 times). Many databases used in Esposito et al. (1997) have missing values. At this stage, we do not consider missing values, and as a result we chose only 4 of the databases considered in Esposito et al. (1997).

The results in Table 5 show that our 2-norm algorithm is competitive to the best algorithms tested in the reference, in both accuracy and size. Our results also show that CCP's and EBP's performance (see Table 5) is close to that presented in Esposito et al. (1997) (the discrepancies are caused by the small size of the databases and the random factor in the partitioning of the databases), so the readers can compare the performance of our algorithm

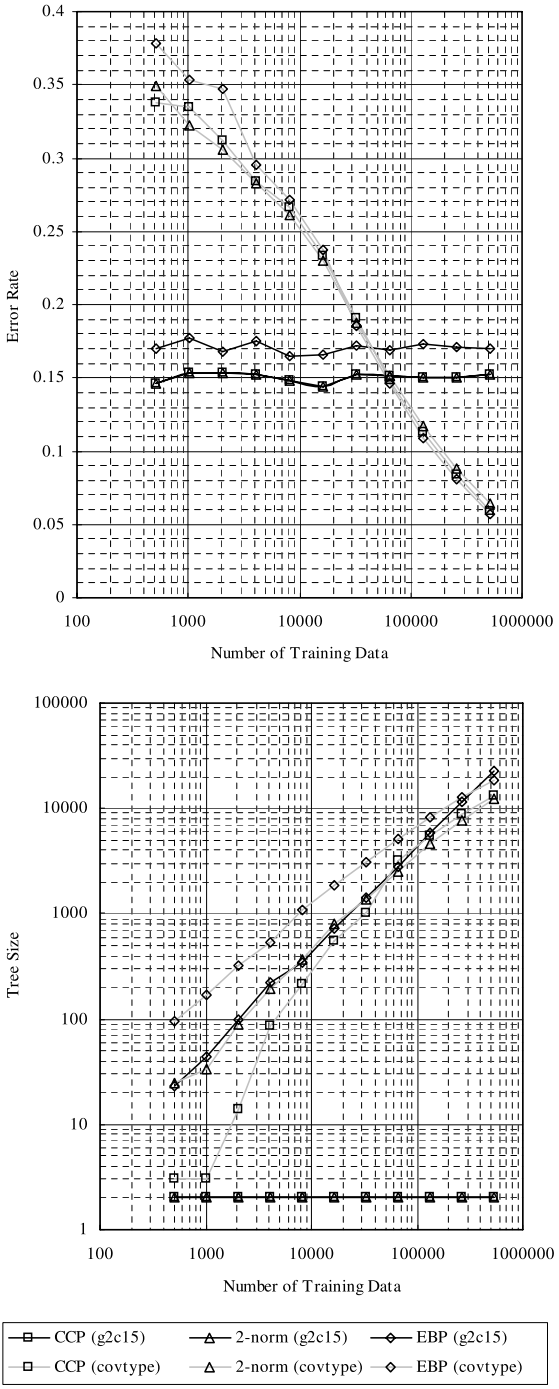
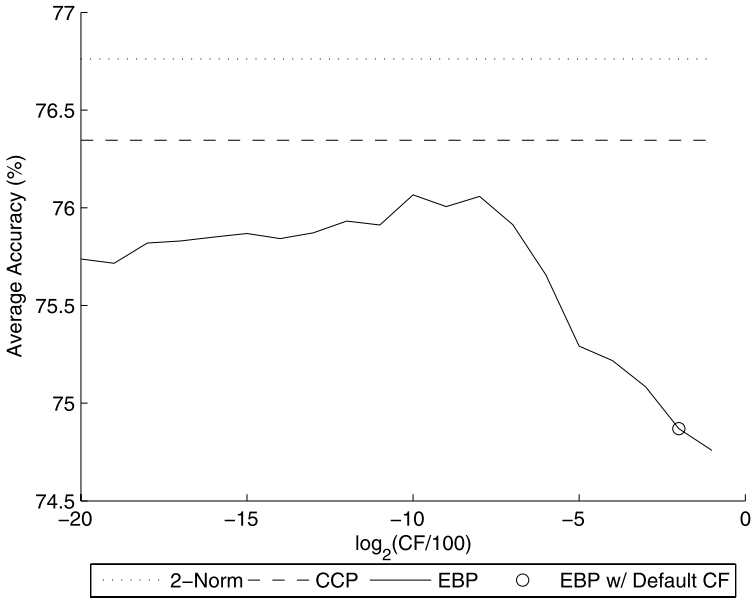
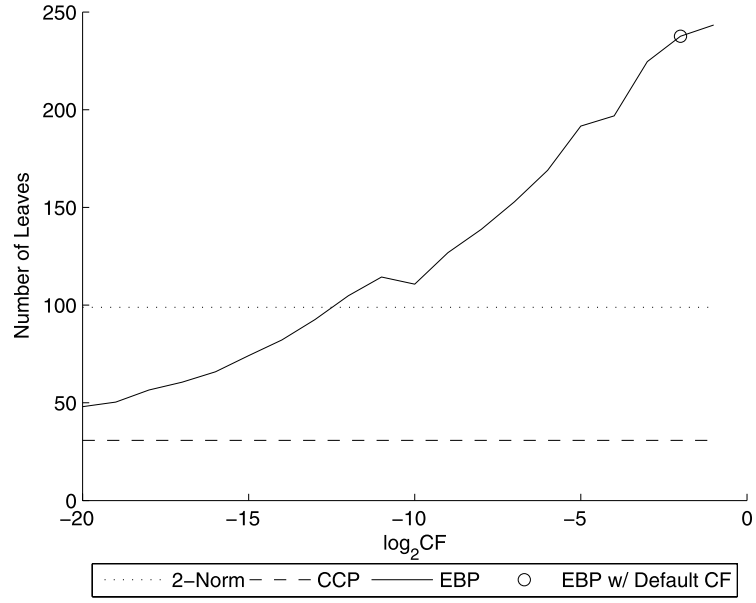


Fig. 5 Error Rate and Tree Size of 2-norm, CCP and EBP on Large Databases (Gaussian Database and *Covertype* Database) as Training Set Size Increases

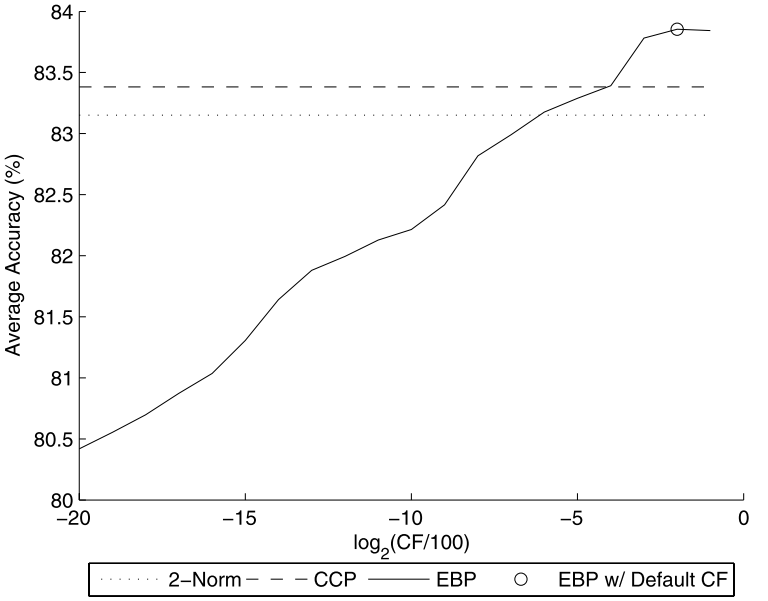


(a) Accuracy on waveform

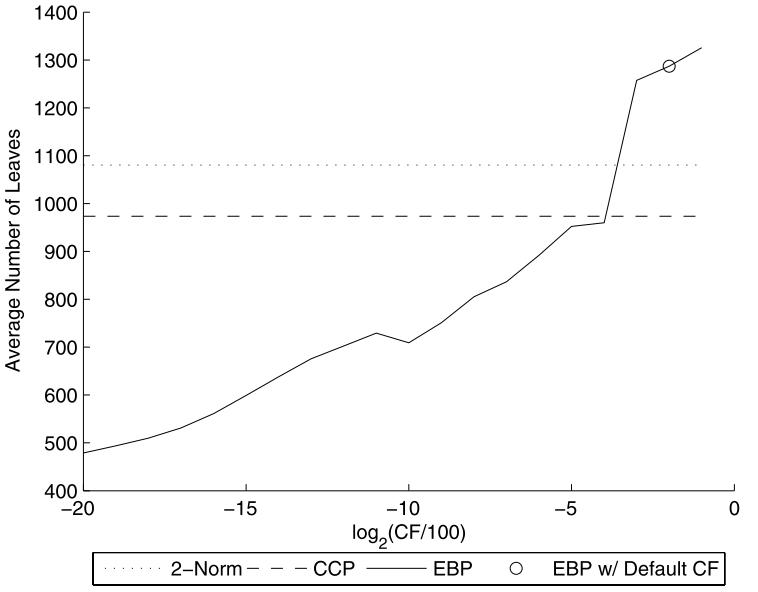


(b) Tree size on waveform

Fig. 6 Comparison of Error Rate and Tree Size among 2-norm, CCP, and EBP with Varied *CF* Value on Databases *Waveform* and *Splice*



(c) Accuracy on splice



(d) Tree size on splice

Fig. 6 (*Continued*)

Table 5 Error Rate (upper half) and Size (lower half) attained by various pruning algorithms (including CCP, 2-norm, and EBP) on small databases

Dataset	CCP	2-norm	EBP	Min Ref	Max Ref	Med Ref
iris	6.04 ± 3.11	6.22 ± 3.32	5.96 ± 3.00	5.07 ± 0.63	11.67 ± 2.62	5.78 ± 0.57
glass	34.13 ± 6.29	32.75 ± 5.04	33.38 ± 5.83	35.31 ± 1.35	41.81 ± 1.86	38.00 ± 1.03
p.gene	22.63 ± 4.36	23.13 ± 5.34	24.63 ± 7.06	21.75 ± 1.40	25.62 ± 2.25	23.88 ± 1.81
blocks	3.24 ± 0.33	3.27 ± 0.37	3.33 ± 0.34	2.98 ± 0.10	3.75 ± 0.16	3.22 ± 0.11
iris	4.44 ± 1.15	3.00 ± 0.00	4.44 ± 0.92	3.13 ± 0.15	5.40 ± 0.27	3.76 ± 0.13
glass	9.76 ± 5.91	12.32 ± 2.37	29.92 ± 2.74	7.04 ± 0.80	28.72 ± 0.58	18.52 ± 1.09
p.gene	4.24 ± 2.02	3.68 ± 1.17	8.52 ± 1.62	4.36 ± 0.47	15.28 ± 0.85	8.44 ± 0.58
blocks	14.04 ± 5.40	20.88 ± 3.63	50.48 ± 4.80	8.08 ± 0.44	78.48 ± 1.28	28.12 ± 2.96

"Min Ref", "Max Ref", and "Med Ref" mean the minimum, the maximum, and the median value, respectively, of the performances of the pruning algorithms reported in Esposito et al. (1997)

given in this paper to the performance of the other pruning algorithms given in Esposito et al. (1997).

6 Summary/conclusions

In this paper, we proposed a k-norm pruning algorithm based on the tree error rate estimation that takes into account both the expected value and the standard deviation of the error. We provided a detailed derivation and interpretation for estimating the mean and standard deviation of the error rate at each tree node, in a recursive fashion, on which estimation the k-norm error pruning algorithm relies.

We performed a thorough and empirical comparison of our k-norm pruning algorithm (for $k = 2$) against two other well-established pruning algorithms (CCP and EBP). The results showed that 2-norm is better in accuracy than CCP, especially for small training set sizes, and it is better than EBP for small or medium training sizes. The results also show the 2-norm algorithm creates smaller trees than EBP, while at times it creates smaller and other times larger trees than CCP. One of the most pronounced advantages of the 2-norm algorithm compared to CCP and EBP is that it requires less time to produce the pruned trees, because it does not need to utilize additional data (training or validation) during pruning. In particular, our experimental results demonstrated that 2-norm pruning is orders of magnitude faster than CCP and EBP. More specifically, experiments with very large databases show that 2-norm pruning scales better than EBP or CCP as the database size increases.

Finally, it is important to note that our approach can be extended by generalizing Theorem 1 to other measures of interests like appropriate misclassification costs. Moreover, k-norm error estimation can be applied for pruning in other classification models, whose design involves partitioning of the input space.

Appendix A: Case study on *Iris* database

The *Iris* database is extensively examined in the literature. It can be downloaded from the UCI repository (Newman et al. 1998). For visualization purposes, we only use two attributes

(petal length and petal width in cm), which have the most significant correlation to the classes. Figure 7(a) shows a scatter plot of these two attributes for the *Iris* data. Each class contains 50 examples, some of which are overlapped.

In this section we demonstrate how to apply the 2-norm estimation to the tree pruning and the tree prediction with Iris database.

A.1 Tree construction

We used CART to grow a fully grown tree with the Gini Index as the impurity measure and using Maximum Likelihood Estimation for the probability estimation. If we use the entropy or the twoing rule to measure the split gain, we get exactly the same tree. The splits of the fully grown tree are shown in Fig. 7(a). Note that an example of class versicolor (circle marker) have exactly the same attributes as an example of class virginica (star marker). CART attempted to separate the former from the class virginica by a vertical split, but then it found out that no split could be found to handle the impure left region, and thus the previous split remains as an ineffective one (a split that is not effective; see Theorem 2 for the definition of effective splits).

A.2 Tree pruning

Both CART's pruning (using 10-fold cross validation and 1-SE rule) and C4.5's pruning remove only the ineffective split. Now we focus on the error rate and apply our 2-norm pruning algorithm. For simplicity, $\lambda = \eta = 0.5$. The optimal pruned tree is shown in Fig. 7(b) and Fig. 8.

The optimal pruned tree can be evaluated as follows. For the leaf c_{21},

$$E[r_{c_{21}}] = \frac{5 + 0.5 \times 2}{54 + 0.5 \times 3} = 0.1081, \tag{51}$$

$$E[r_{c_{21}}^2] \approx \frac{6 \times 7}{55.5 \times 56.5} = 0.01339, \tag{52}$$

$$P^*[A_{c_{21}}|A_{c_2}] = \frac{54 + 0.5}{100 + 0.5 \times 2} = 0.5396. \tag{53}$$

For the leaf c_{22},

$$E[r_{c_{22}}] = \frac{1 + 0.5 \times 2}{46 + 0.5 \times 3} = 0.04211, \tag{54}$$

$$E[r_{c_{22}}^2] \approx \frac{2 \times 3}{47.5 \times 48.5} = 0.002604, \tag{55}$$

$$P^*[A_{c_{22}}|A_{c_2}] = \frac{46 + 0.5}{100 + 0.5 \times 2} = 0.4604. \tag{56}$$

For the decision node c_2 (note that the event A_{root} is always true, and thus $P^*[A_{c_2}|A_{root}] = P^*[A_{c_2}]$),

$$E[r_{c_2}] = E[r_{c_{21}}]P^*[A_{c_{21}}|A_{c_2}] + E[r_{c_{22}}]P^*[A_{c_{22}}|A_{c_2}] = 0.07772, \tag{57}$$

$$E[r_{c_2}^2] = E[r_{c_{21}}^2]P^*[A_{c_{21}}|A_{c_2}] + E[r_{c_{22}}^2]P^*[A_{c_{21}}|A_{c_2}] \approx 0.008427, \tag{58}$$

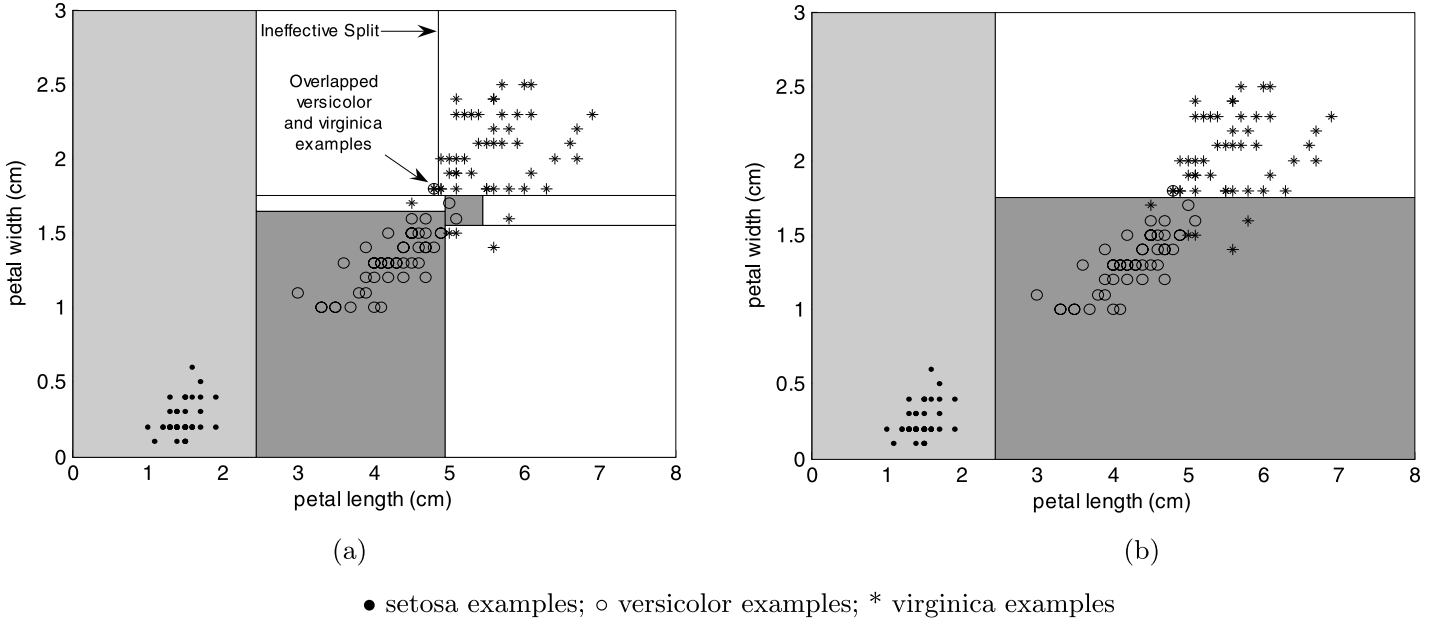


Fig. 7 Fully Grown Tree and Pruned Tree for Iris data. (**a**) Fully grown tree for the Iris problem; input attributes to the tree are only the Iris petal length and Iris petal width; CART with Gini criterion is used to grow the tree. (**b**) Optimally pruned tree for the Iris problem, using as a criterion the 2-norm error rate for pruning a tree node

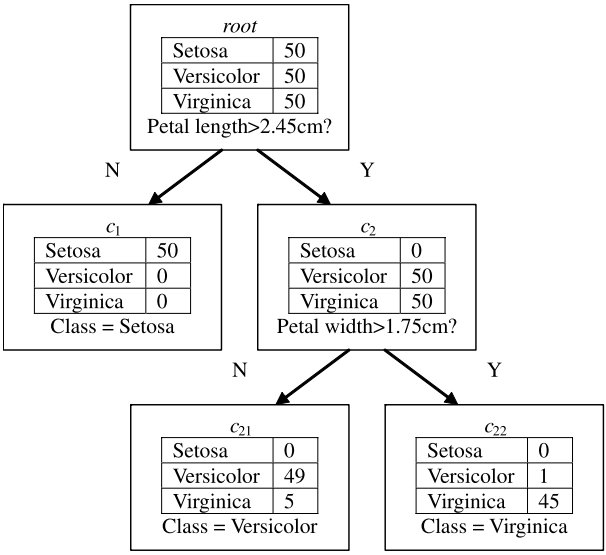


Fig. 8 The optimally pruned tree for the Iris problem when the 2-norm error rate is used as the criterion to prune tree nodes. In this figure, the numbers of examples of each Iris type residing at every node of the tree is shown. Furthermore, on top of every link (connecting two tree nodes) the criterion used to split the data at the node is shown. Tree nodes are designated as *root*, or c; when the c designation is used to denote a node the associated subscripts indicate the ancestry (e.g., c_{21} means the first child of the second child of the root)

$$P^*[A_{c_2}] = \frac{100 + 0.5}{150 + 0.5 \times 2} = 0.6656. \tag{59}$$

For the leaf c_1,

$$E[r_{c_1}] = \frac{0 + 0.5 \times 2}{50 + 0.5 \times 3} = 0.01942, \tag{60}$$

$$E[r_{c_1}^2] \approx \frac{1 \times 2}{51.5 \times 52.5} = 0.0007397, \tag{61}$$

$$P^*[A_{c_1}] = \frac{50 + 0.5}{150 + 0.5 \times 2} = 0.3344. \tag{62}$$

For the root (representing the entire tree)

$$E[r_{root}] = E[r_{c_1}]P^*[A_{c_1}] + E[r_{c_2}]P^*[A_{c_2}] = 0.05822, \tag{63}$$

$$E[r_{root}^2] = E[r_{c_1}^2]P^*[A_{c_1}] + E[r_{c_2}^2]P^*[A_{c_2}] \approx 0.005856, \tag{64}$$

$$\sigma[r_{root}] = \sqrt{E[r_{root}^2] - E[r_{root}]^2} \approx 0.04966, \tag{65}$$

$$\|r_{root}\|_2 = \sqrt{E[r_{root}^2]} = 0.07652, \tag{66}$$

where $E[r_{root}]$ is the expected value of the error rate and the standard deviation 0.04966 measures the uncertainty of the error rate; $\|r_{root}\|_2$ gives a 2-norm estimation of the error rate (combining the expected value and the standard deviation). These measures are defined for the average case. When an input $\mathbf{X}$ is given, we have more specific measures that depend on $\mathbf{X}$, as shown in the next subsection.

A.3 Tree prediction

We use the tree pruned with $k = 2$. If an unseen datum falls in the leaf c_1 (left sub-region), the predicted class is "setosa," with the predicted error rate and the corresponding reliability computed as follows:

$$E_{\mathbf{P}}[r] \approx E[r_{c_1}] = 0.01942, \tag{67}$$

$$\sigma_{\mathbf{P}}[r] \approx \sqrt{E[r_{c_1}^2] - E[r_{c_1}]^2} = 0.0190. \tag{68}$$

To combine the standard deviation into the estimate of the error rate, we can use the 2-norm:

$$\|r\|_2 \approx \sqrt{E[r_{c_1}^2]} = 0.0272. \tag{69}$$

Similarly, if an unseen datum falls in the leaf c_{21} (bottom-right sub-region), the predicted class is "versicolor," with the following predicted error rate:

$$E_{\mathbf{P}}[r] \approx E[r_{c_{21}}] = 0.1081, \tag{70}$$

$$\sigma_{\mathbf{P}}[r] \approx \sqrt{E[r_{c_{21}}^2] - E[r_{c_{21}}]^2} = 0.04131, \tag{71}$$

$$\|r\|_2 \approx \sqrt{E[r_{c_{21}}^2]} = 0.1157. \tag{72}$$

If an unseen datum falls in the leaf c_{22} (top-right sub-region), the predicted class is "virginica," with the following predicted error rate:

$$E_{\mathbf{P}}[r] \approx E[r_{c_{22}}] = 0.04211, \tag{73}$$

$$\sigma_{\mathbf{P}}[r] \approx \sqrt{E[r_{c_{22}}^2] - E[r_{c_{22}}]^2} = 0.02884, \tag{74}$$

$$\|r\|_2 \approx \sqrt{E[r_{c_{22}}^2]} = 0.05103. \tag{75}$$

Appendix B: Proof of Theorem 1

Our goal is to prove the following equation for any decision node d and any event Φ:

$$E_{\mathbf{X}|A_d}[r_d^k] = \sum_{c \in Children(d)} E_{\mathbf{X}|A_c}[r_c^k] P[A_c|A_d], \tag{76}$$

where

$$r_d = P[\Phi|A_d, \mathbf{X}], \tag{77}$$

given that the events $\{A_c|A_d\}_{c \in Children(d)}$ are mutually exclusive and collectively exhaustive and that $P[A_c|A_d, \mathbf{X}]$ is either zero or one for any $\mathbf{X}$.

Proof Recall that for any $c \in Children(d)$, A_c implies A_d, which means $A_c = (A_c, A_d)$. As a result,

$$P[\Phi|A_d, \mathbf{X}] = \sum_{c \in Children(d)} P[\Phi|A_c, \mathbf{X}] P[A_c|A_d, \mathbf{X}], \tag{78}$$

that is,

$$r_d = \sum_{c \in Children(d)} r_c P[A_c|A_d, \mathbf{X}]. \tag{79}$$

Since $P[A_c|A_d, \mathbf{X}]$ is one for one of the children and is zero for the other children,

$$P[A_{c_1}|A_d, \mathbf{X}]P[A_{c_2}|A_d, \mathbf{X}] = 0, \quad \text{if } c_1 \neq c_2, \tag{80}$$

$$P[A_c|A_d, \mathbf{X}]^k = P[A_c|A_d, \mathbf{X}], \tag{81}$$

and therefore,

$$r_d^k = \sum_{c \in Children(d)} r_c^k P[A_c|A_d, \mathbf{X}], \tag{82}$$

$$\begin{aligned} E_{\mathbf{X}|A_d}[r_d^k] &= \int r_d^k f(\mathbf{X}|A_d) d\mathbf{X} \\ &= \sum_{c \in Children(d)} \int [r_c^k] P[A_c|A_d, \mathbf{X}] f(\mathbf{X}|A_d) d\mathbf{X}. \end{aligned} \tag{83}$$

According to Bayesian Theory,

$$P[A_c|A_d, \mathbf{X}] f(\mathbf{X}|A_d) = f(\mathbf{X}|A_c, A_d) P[A_c|A_d] = f(\mathbf{X}|A_c) P[A_c|A_d], \tag{84}$$

and therefore,

$$\begin{aligned} E_{\mathbf{X}|A_d}[r_d^k] &= \sum_{c \in Children(d)} \int [r_c^k] f(\mathbf{X}|A_c) d\mathbf{X} P[A_c|A_d] \\ &= \sum_{c \in Children(d)} E_{\mathbf{X}|A_c}[r_c^k] P[A_c|A_d]. \end{aligned} \tag{85}$$

□

Appendix C: Proof of Theorem 2

The 1-norm error rate of the node t before the split is

$$r_{leaf}(t) = \frac{b_t + (J-1)\lambda}{n_t + J\lambda}, \tag{86}$$

where b_t is the number of training examples in t of minority classes (that is, the number of misclassified training examples in t), and $b_t = n_t - \max_j n_{j,t}$. After the split,

$$r_{tree}(t) = \sum_{c \in Children(t)} r_{leaf}(c) \frac{n_c + \eta}{n_t + K_t \eta}. \tag{87}$$

Our goal is to prove $r_{tree}(t) < r_{leaf}(t)$ under the following assumptions:

1. The *Determinacy Assumption*. Under this assumption, when the number of misclassifications is decreased, the decrease is at least one, that is, $\sum_{c \in Children(t)} b_c \leq b_t - 1$.
2. $0 \leq \eta \leq \lambda J$.

3. $\lambda \leq \frac{1}{(K_t-1)(J-1)}$.

Proof Note that $r_{leaf}(t)$ is independent of η. We first find the maximum value of $r_{tree}(t)$ with respect to η

$$\frac{\partial}{\partial \eta} \sum_c \frac{(b_c + (J-1)\lambda)(n_c + \eta)}{(n_c + J\lambda)(n_t + K_t\eta)} = \sum_c \frac{(b_c + (J-1)\lambda)(n_t - K_t n_c)}{(n_c + J\lambda)(n_t + K_t\eta)^2}$$
$$= \frac{q}{(n_t + K_t\eta)^2}. \tag{88}$$

Note that q is independent of η and thus the sign of the above derivative is independent of η, which means that the maximum value is achieved either when $\eta = 0$ or when $\eta = J\lambda$. It suffices to prove the theorem for these two cases.

Case $\eta = 0$:

$$\begin{aligned} r_{tree}(t) &= \frac{1}{n_t} \sum_{c \in Children(t)} \frac{n_c(b_c + (J-1)\lambda)}{n_c + J\lambda} \\ &< \frac{1}{n_t} \sum_{c \in Children(t)} \frac{n_t(b_c + (J-1)\lambda)}{n_t + J\lambda} \\ &\leq \frac{b_t + K_t(J-1)\lambda - 1}{n_t + J\lambda} \\ &\leq \frac{b_t + K_t(J-1)\lambda - (K_t - 1)(J-1)\lambda}{n_t + J\lambda} \\ &= \frac{b_t + (J-1)\lambda}{n_t + J\lambda} \\ &= r_{leaf}(t). \end{aligned} \tag{89}$$

Case $\eta = J\lambda$:

$$\begin{aligned} r_{tree}(t) &= \frac{\sum_{c \in Children(t)}(b_c + (J-1)\lambda)}{n_t + K_t J\lambda} \\ &\leq \frac{b_t - 1 + (J-1)K_t\lambda}{n_t + K_t J\lambda} \\ &< \frac{b_t}{n_t + J\lambda} + \frac{(J-1)K_t\lambda - 1}{n_t + K_t J\lambda}. \end{aligned} \tag{90}$$

Given $0 \leq \lambda \leq \frac{1}{(J-1)(K_t-1)}$, it is not difficult to prove that

$$\frac{(J-1)K_t\lambda - 1}{n_t + K_t J\lambda} < \frac{(J-1)\lambda}{n_t + J\lambda}, \tag{91}$$

and thus $r_{tree}(t) < r_{leaf}(t)$. □

Acknowledgements This work was supported in part by the NSF grants: 0203446, 0341601, 05254209, 0647120, 0647018, 0717680, and 0717674.

References

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Belmont: Wadsworth.

Cestnik, B., & Bratko, I. (1991). On estimating probabilities in tree pruning. In *EWSL-91: Proceedings of the European working session on learning on machine learning* (pp. 138–150). New York: Springer.

Chernoff, H. (1952). A measure of asymptotic efficiency of tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, *23*, 493–507.

Dong, M., & Kothari, R. (2001). Classifiability based pruning of decision trees. In *IJCNN* (Vol. 3, pp. 1739–1743).

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern classification* (2nd ed.). Hardcover: Wiley-Interscience.

Elomaa, T., & Kääriäinen, M. (2001). An analysis of reduced error pruning. *Journal of Artificial Intelligence Research*, *15*, 163–187.

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

Freund, Y. (1998). Self bounding learning algorithms. In *COLT: Proceedings of the workshop on computational learning theory*. San Mateo: Morgan Kaufmann.

Good, I. J. (1965). *The estimation of probabilities: an essay on modern Bayesian methods (No. 30)*. Cambridge: MIT Press.

Good, I. J. (1967). A Bayesian significance test for multinomial distributions. *Journal of the Royal Statistical Society, Series B (Methodological)*, *29*(3), 399–431.

Hall, L. O., Bowyer, K. W., Banfield, R. E., Eschrich, S., & Collins, R. (2003). Is error-based pruning redeemable? *International Journal on Artificial Intelligence Tools*, *12*(3), 249–264.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, *58*, 13–30.

Kääriäinen, M., & Elomaa, T. (2003). Rademacher penalization over decision tree prunings. In N. Lavrac, D. Gamberger, L. Todorovski, & H. Blockeel (Eds.), *Lecture notes in computer science* (Vol. 2837, pp. 193–204). Berlin: Springer.

Kääriäinen, M., Malinen, T., & Elomaa, T. (2004). Selective Rademacher penalization and reduced error pruning of decision trees. *Journal of Machine Learning Research*, *5*, 1107–1126.

Kearns, M. J., & Mansour, Y. (1998). A fast, bottom-up decision tree pruning algorithm with near-optimal generalization. In *ICML '98: proceedings of the fifteenth international conference on machine learning* (pp. 269–277). San Francisco: Morgan Kaufmann.

Kijsirikul, B., & Chongkasemwongse, K. (2001). Decision tree pruning using backpropagation neural networks. In *IJCNN* (Vol. 3, pp. 1876–1880).

Kohavi, R., Becker, B., & Sommerfield, D. (1997). Improving simple Bayes. In M. van Someren & G. Widmer (Eds.), *Lecture notes in computer science* (Vol. 1224, pp. 78–87). Berlin: Springer.

Krichevskiy, R. E. (1998). Laplace's law of succession and universal encoding. *IEEE Transactions on Information Theory*, *44*(1), 296–303.

Lidstone, G. J. (1920). Note on the general case of the Bayes-Laplace formula for inductive or a posteriori probabilities. *Transactions of the Faculty of Actuaries*, *8*, 182–192.

Mansour, Y. (1997). Pessimistic decision tree pruning based on tree size. In *Proceedings of the 14th international conference on machine learning* (pp. 195–201). San Mateo: Morgan Kaufmann.

Mansour, Y., & McAllester, D. A. (2000). Generalization bounds for decision trees. In *COLT '00: proceedings of the thirteenth annual conference on computational learning theory* (pp. 69–74). San Francisco: Morgan Kaufmann.

Mehta, M., Rissanen, J., & Agrawal, R. (1995). MDL-based decision tree pruning. In *KDD* (pp. 216–221).

Newman, D. J., Hettich, S., Blake, C. L., & Merz, C. J. (1998). *UCI repository of machine learning databases*. Available from http://www.ics.uci.edu/~mlearn/MLRepository.html.

Niblett, T., & Bratko, I. (1986). Learning decision rules in noisy domains. In *Proceedings of expert systems '86, the 6th annual technical conference on research and development in expert systems III* (pp. 25–34).

Quinlan, J. R. (1993). *C4.5: programs for machine learning*. San Mateo: Morgan Kaufmann.

Quinlan, J. R. (1999). Simplifying decision trees. *International Journal of Human-Computer Studies*, *51*(2), 497–510.

Ristad, E. S. (1995). *A natural law of succession* (Tech. Rep. No. TR-495-95). Princeton University.

Stone, M. (1978). Cross-validation: a review. *Mathematics, Operations and Statistics*, *9*, 127–140.
Vardeman, S. B., & Jobe, J. M. (2001). *Basic engineering data collection and analysis*. Brooks/Cole Thompson Learning.
Windeatt, T., & Ardeshir, G. (2001). An empirical comparison of pruning methods for ensemble classifiers. In *IDA '01: proceedings of the 4th international conference on advances in intelligent data analysis* (pp. 208–217). London: Springer.
Zemanian, A. (1987). *Distribution theory and transform analysis: an introduction to generalized functions, with applications*. New York: Dover.