An Analysis Of Misclassification Rates For Decision Trees

Mingyu Zhong
University of Central Florida

An Analysis of Misclassification Rates for Decision Trees

by

Mingyu Zhong
钟鸣宇
B.S. Tsinghua University, 2002
M.S. University of Central Florida, 2005

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the School of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2007

Major Professors:

Michael Georgiopoulos
Georgios C. Anagnostopoulos

Abstract

The decision tree is a well-known methodology for classification and regression. In this dissertation, we focus on the minimization of the misclassification rate for decision tree classifiers. We derive the necessary equations that provide the optimal tree prediction, the estimated risk of the tree's prediction, and the reliability of the tree's risk estimation. We carry out an extensive analysis of the application of Lidstone's law of succession for the estimation of the class probabilities. In contrast to existing research, we not only compute the expected values of the risks but also calculate the corresponding reliability of the risk (measured by standard deviations). We also provide an explicit expression of the k-norm estimation for the tree's misclassification rate that combines both the expected value and the reliability. Furthermore, our proposed and proven theorem on k-norm estimation suggests an efficient pruning algorithm 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 that compares very favorably with two other well-known pruning algorithms, CCP of CART and EBP of C4.5. Finally, our work provides a deeper understanding of decision trees.

To my wife Min Hu and my parents Junxing Zhong and Shaozhen Hu

谨此献给我的爱人胡敏，及我的父母钟均行、胡韶珍

Acknowledgments

I would like to take the opportunity to express my gratitude to a number of individuals who have helped me during my doctoral research. Especially, I would like to thank my advisor Dr. Michael Georgiopoulos for his generous time and commitment. He has stimulated my efforts with his many suggestions since the very beginning of my research. He has also insisted with relentless encouraging the publication of my research in conference and journal venues. Moreover, I appreciate the generous help from my co-advisor, Dr. Georgios C. Anagnostopoulos, who tirelessly offered his advice on many occasions and discussions that we have had about my research efforts; he is the scholar who always pushes me to attain higher in my research endeavors. I would also like to extend my thanks to my other dissertation committee members, Drs. A. Gonzalez, R. DeMara, and A. S. Wu, who have devoted their time and effort to provide their helpful suggestions during my candidacy and dissertation oral examinations.

I extend many thanks to the editorial board and the anonymous reviewers of *Machine Learning* journal for their constructive comments in more effectively presenting the ideas of my research.

Additionally, I would like to express my gratitude to my Machine Learning Lab colleagues, Jimmy Secretan, Chris Sentelle, and Anna Koufakou, who continuously helped me in my daily work.

Finally, I am extremely grateful to my wife, Min Hu, as well as my parents, who have always been supporting and understanding. This dissertation would not be possible if it were not for their consistent and unconditional support.

This dissertation acknowledges the support by the National Science Foundation grants: CRCD 0203446, CCLI 0341601, DUE 05254209, IIS-REU 0647120, and IIS-REU 0647018.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

Prediction and decision making are frequently used in real world applications. To aid humans in these tasks, the field of Machine Learning provides a variety of algorithms for automatic learning and prediction/classification. Most of the automatic predictors/classifiers can optimize their output based on a certain minimal risk criterion. In many modern applications, however, the optimized output alone is not sufficient to solve our problems. In making decisions, sometimes we need to know not only the best decision (the one with the least risk) but also the associated risk. For example, in classification problems the risk can be defined as the misclassification rate. Since the risk is seldom deterministic, we might also need the reliability of the risk estimation. For example, compare the following outputs regarding the class of an input that is presented to the classifier.

1. Class 1 has the lowest misclassification rate (or highest class probability)

2. Class 1 has the highest class probability that is approximately 0.6

3. Class 1 has the highest class probability that lies in the range 0.6 ± 0.1

Obviously, the last classifier gives the most complete information among the three.

In this dissertation we focus on decision tree classifiers. The advantages of decision trees include their time efficiency in training, their compatibility with categorical (nominal) inputs and data with missing values, and the interpretability (to humans) of their knowledge representation. Their accuracy is also comparable to other Machine Learning algorithms, such as neural networks (see [LLS00]).

Through the analysis conducted in this dissertation we are not only able to provide the optimal tree prediction, but we are also able to calculate the reliability of this prediction. Furthermore, we propose a pruning algorithm, referred to as k-norm pruning algorithm, which has the following properties: it has a clear theoretical interpretation, 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 it is simple and easy to implement. Furthermore, it compares favorably with two well known pruning strategies the Cost Complexity Pruning (CCP) of CART (see [BFO84]), and the Error Based Pruning (EBP) of C4.5 (see [Qui93]).

The rest of this dissertation is organized as follows. In Chapter 2, we first discuss the pros and cons of some 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 Chapter 3, which introduces the reader to decision trees and probability estimation. We elaborate on the theoretical analysis of the tree predictions in Chapter 4. In particular, we apply Lidstone's law of succession (see [Lid20]) 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 to maximize the estimated accuracy, with the properties mentioned earlier. In Chapter 5 we further show a series of useful properties of k-norm pruning. In Chapter 6, we show the experimental results of our pruning algorithm by comparing it to two other classical algorithms (CCP and EBP), as well as other pruning algorithms. We emphasize the advantages of the k-norm approach compared to alternative pruning methods, such as CCP and EBP. We also extend our k-norm estimation to two other risk definitions in classification problems, misclassification costs and error of class probabilities, in Chapter 7. We summarize our work in Chapter 8. In the appendices, we provide a list of frequently used notations and a detailed example of our estimation approach to a well studied problem, as well as the proofs of our theorems.

CHAPTER 2
MOTIVATION OF OUR WORK

In this chapter we take a careful look at current, well-known estimation approaches, and we provide the motivation for our work.

2.1 Tree Pruning

Hyafil and Rivest proved that getting the optimal tree is NP-complete (see [HR76]), and thus most algorithms employ the greedy search and the divide-and-conquer approach to grow a tree. Like other approaches that design 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, such as the Minimal Cost-Complexity Pruning (CCP) in CART (Classification and Regression Trees; see [BFO84], page 66), Error Based Pruning (EBP) in C4.5 (see [Qui93], page 37), Minimum Error Pruning (MEP) (see [NB86, CB91]), Reduced Error Pruning (REP), Pessimistic Error Pruning (PEP) (see [Qui99] for both REP and PEP), the MDL-Based Pruning (see [MRA95]), Classifiability Based Pruning (see [DK01]), the pruning using Backpropagation Neural Networks (see [KC01]), etc. Some of the pruning algorithms are briefly analyzed and empirically compared in [EMS97]. Our basis of comparison of the merits of the proposed k-norm pruning algorithm are the benchmark pruning algorithms, CCP and EBP. However some comparisons are also made between k-norm pruning and the other pruning algorithms, mentioned in the above references.

It would be very easy to estimate the error rate on unseen data if a separate validation set with sufficient size is given. When only the training set is given, one could separate the training set into two parts, one for growing the tree and one for validating the tree (for example, see [Qui99] for REP and [KE03, KME04] for its two successors); 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 (see [Sto78]), which has been successfully applied in CCP. Nevertheless, validation is usually a computationally expensive procedure. Most of the estimation approaches that do not depend on validation can be divided into two categories: maximum likelihood estimation and posterior (Bayesian) estimation. They are discussed in the next two sections.

2.2 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 (see [Hoe63]), Chernoff's Inequality (see [Che52]), or the Binomial confidence interval. For example, the following can be derived from Hoeffding's Inequality:

> If $X_1, X_2, \ldots, X_n$ are independent random variables distributed in the range [0,1], then
> $$P\left[\frac{\sum_{i=1}^{n} X_i}{n} - \frac{E\left[\sum_{i=1}^{n} X_i\right]}{n} \geq \varepsilon\right] \leq e^{-2n\varepsilon^2}. \tag{2.1}$$

In a tree node, many researchers define X_i as the indicator function of the event that the i-th training example is correctly classified (1 if correct or 0 if incorrect). Under this definition, they interpret $\frac{\sum_{i=1}^{n} X_i}{n}$ as the maximum likelihood estimate of the accuracy (or observed accuracy) and $\frac{E\left[\sum_{i=1}^{n} X_i\right]}{n}$ as the true accuracy. The above inequality (2.1) yields a practical bound of the generalization accuracy. In [Qui93, Man97, Fre98, KM98, MM00, KE03], the authors have experimented with different ε values and/or applying other inequalities to achieve better confidence bounds/intervals for the generalization accuracy. Note that in EBP (see [Qui93]), the equation for computing the estimated error rate is not shown; it can be found in [WA01] and turns out to be a one-sided binomial 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 researchers have also proposed tree pruning algorithms 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 [VJ01], page 342. For instance, if through the observation of 1000 values (X_i's) for the random variable X (1 if correct or 0 if incorrect) using 1000 training examples, we come up with an average accuracy $\overline{X} = \frac{\sum_{i=1}^{1000} X_i}{1000}$ (maximum likelihood estimate of the accuracy), and if $\varepsilon = 0.05$, we get $e^{-2n\varepsilon^2} = 0.0067$. In this example, Hoeffding's inequality simply implies that if we repeat the experiment for a sufficiently large number of times, and in the k-th run we observe the average accuracy $\overline{X}^k$ using 1000 independent training examples of the same distribution, then in at least 99.33% ($= 1 - 0.0067$) of the runs, the actual expected value $E[X]$ is higher than $\overline{X}^k - 0.05$. For any particular k, it does not imply that $E[X]$ is higher than $\overline{X}^k - 0.05$ at a certain probability, because $E[X]$ and $\overline{X}^k - 0.05$ are both deterministic. In other words, 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.

Secondly, there is another difficulty in comparing the estimated error rates before and after pruning a sub-tree. In [Man97, KM98, MM00], the authors apply an upper bound to estimate error rate of a sub-tree, but simply compare it to the maximum likelihood estimate of the (pruned) leaf error rate rather than another upper bound. That is, the penalty applies only to the tree, which makes the comparison unfair. On the other hand, if a similar penalty applies to the leaf, this approach tends to under-prune the tree, because in the sub-tree the observed misclassifications are always fewer than in the leaf while the total instances are the same. In EBP (see [Qui93]) and two improved versions of REP (see [KE03, KME04]), the authors use a more reasonable strategy: the number of misclassifications of a tree is estimated as the sum of those in its leaves. Then, the misclassification at a leaf is estimated by using a confidence interval. Since in the leaves the penalty might be relatively significant, because fewer examples are observed, the total penalty may result in the decision of pruning. However, the comparison is still questionable. For example, if e_1 and e_2 are 90% confidence upper bounds of E_1 and E_2 (the number of misclassifications made in the left child and in the right child, respectively), that is, even if $P[E_1 \leq e_1] = 0.9$ and $P[E_2 \leq e_2] = 0.9$ (ignoring the misinterpretation mentioned above), generally $P[E_1 + E_2 \leq e_1 + e_2] \neq 0.9$, which means that the sum $e_1 + e_2$ is not necessarily the 90% confidence upper bound of the sub-tree. Consequently, it might not be fair to compare this sum to a 90% confidence upper bound of the error rate of the pruned node. In summary, we have not found in the literature, a bounding method that can be applied to both leaves and decision nodes consistently.

To the best of our knowledge, we have not seen any statistically significant (at least 1000 points in the database) experimental results 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 [EMS97]. The behavior of EBP can be explained with the theoretical analysis in [EK01], 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. One could make EBP prune more nodes by using extremely small values of the parameter CF as in [HBB03], where $CF = 0.01$ is shown to produce a better tree than the default value $CF = 25$. In this case, however, the estimated error rate loses its true meaning: for a leaf that has 1 misclassification out of 99 (this example is discussed in section 4.3 of this dissertation), the error rate estimate is as pessimistic as 15.8% for $CF = 0.01$, in contrast to 2.6% for $CF = 25$.

2.3 Posterior Estimation

Contrary to maximum likelihood estimation, posterior estimations assume that unknown parameters θ are random variables, and model them by an posterior joint probability density function (abbreviated as PDF; denoted by f) based on Bayesian theorem:

$$f(\theta|\mathbf{O}) = \frac{f(\mathbf{O}|\theta)f(\theta)}{f(\mathbf{O})} = \frac{f(\mathbf{O}|\theta)f(\theta)}{\int_\theta f(\mathbf{O}|\theta)f(\theta)d\theta}, \tag{2.2}$$

where θ is the hypothesis and $\mathbf{O}$ is the observation. When θ stands for one or more probabilities and the observation is the result of a series of Bernoulli trials, $f(\mathbf{O}|\theta)$ is not difficult to compute (see (3.17)). The apparent difficulty is the unknown prior PDF $f(\theta)$. 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 (see [NB86]), where Lidstone's law of succession (see [Lid20]) is used (see section 3.2 for details). Lidstone's law of succession can be derived by choosing the prior joint PDF to be represented by the Dirichlet distribution (3.13) and by choosing the posterior expected value as the estimated value. The prior PDF should model the class probabilities rather than the misclassification rate (see section 3.2 for details). In this dissertation, we prove that the comparison between the estimated error rates before and after pruning is fair (see Theorem 4.1 in section 4.2). However, we also prove that MEP tends to under-prune a tree as well: 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 4.3 in section 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 section 4.3 for a quantitative example).

2.4 k-Norm Estimate

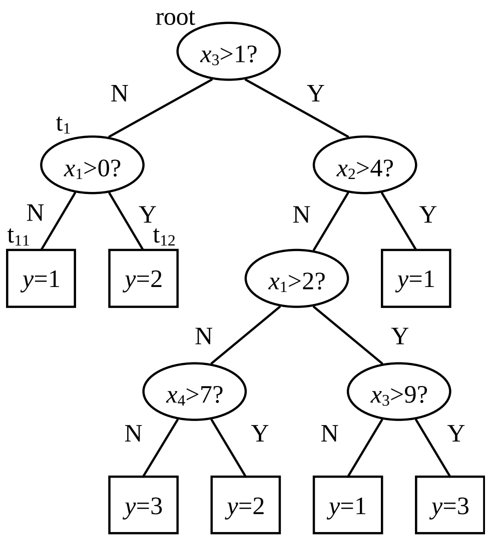
In our work we choose to prune a tree by minimizing the estimated error rate, because we desire not only the optimal pruned tree but also the performance prediction. In our work, we choose not to rely on a validation set because validation may not be practical when the training set is small and it is expensive 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 results in Lidstone's law of succession, 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, which is a generalization of the 2-norm error rate pruning (that relies on estimation of averages and standard deviations of error rates; see section 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).

CHAPTER 3
PRELIMINARIES

This chapter provides some basic information on decision trees and probability estimation approaches, needed for the rest of this dissertation.

3.1 Decision Trees

A typical tree is shown in Figure 3.1.



Note: The variables x_1 to x_4 represent the attributes, and y represents the variable, whose value the tree is predicting. The nodes of the tree are designated by the index t, and the Y, N designations refer to YES or NO answer to the question posed by the node above them.

Figure 3.1: Graph of a Decision Tree

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 (see [BFO84]). In this dissertation, 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 the tree classification from the perspective of posterior probabilities.

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. To view the decision tree classification in a probabilistic way, we can express $P[C_j|\mathbf{X}]$ as

$$P\left[C_j|\mathbf{X}\right] = P\left[C_j|A_{root}, \mathbf{X}\right], \tag{3.1}$$

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 Figure 3.1, $A_{t_1} = (x_3 \leq 1)$ 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 $\{A_c|A_d, c \in Children(d)\}$ are mutually exclusive and collectively exhaustive, where $Children(d)$ is the set of immediate children of node d.

Determinacy Assumption: Given $\mathbf{X}$, the path of X from the root to a leaf is deterministic, that is, $P\left[A_c|A_d, \mathbf{X}\right]$ is either 0 or 1(for example, in Figure 3.1, $P\left[A_{t_{11}}|A_{t_1}, \mathbf{X}\right] = I[x_1 \leq 0]$).

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,

$$P\left[C_j|A_d, \mathbf{X}\right] = \sum_{c \in Children(d)} P\left[C_j|A_c, A_d, \mathbf{X}\right] P\left[A_c|A_d, \mathbf{X}\right]. \tag{3.2}$$

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, (3.2) can be rewritten as

$$P\left[C_j|A_d, \mathbf{X}\right] = \sum_{c \in Children(d)} P\left[C_j|A_c, \mathbf{X}\right] P\left[A_c|A_d, \mathbf{X}\right], \tag{3.3}$$

or equivalently,

$$P\left[C_j|\mathbf{X}\right] = \sum_{l \in Leaves} P\left[C_j|A_l, \mathbf{X}\right] P\left[A_l|\mathbf{X}\right]. \tag{3.4}$$

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

$$P\left[C_j|A_l, \mathbf{X}\right] \approx P\left[C_j|A_l\right]. \tag{3.5}$$

$P\left[C_j|A_l\right]$ is unknown and has to be estimated (e.g., by maximum likelihood estimation or Lidstone's law of succession). Let $P^*\left[C_j|A_l\right]$ denote the estimated value of $P\left[C_j|A_l\right]$ (the asterisk representing the estimated value throughout this dissertation). The predicted class probability $P^*\left[C_j|\mathbf{X}\right]$ is estimated as follows:

$$P^*\left[C_j|\mathbf{X}\right] = \sum_{l \in Leaves} P^*\left[C_j|A_l\right] P\left[A_l|\mathbf{X}\right]. \tag{3.6}$$

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

$$P^*\left[C_j|\mathbf{X}\right] = P^*\left[C_j|A_{l_{\mathbf{X}}}\right]. \tag{3.7}$$

Interestingly, whether the maximum likelihood estimation or the Lidstone's law of succession is applied for computing the estimated value $P^*\left[C_j|A_l\right]$ (the asterisk representing the estimated value throughout this dissertation),

$$Label^*(\mathbf{X}) = Label(l_{\mathbf{X}}) = \arg\max_j P^*\left[C_j|A_{l_{\mathbf{X}}}\right] = \arg\max_j n_{j,l_{\mathbf{X}}}, \tag{3.8}$$

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 the theorem 1 and corollary 1, based on which the k-norm estimation algorithm is derived (see section 4.2 and section 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 (3.6) includes two approximations:

$$P\left[C_j|\mathbf{X}\right] = \sum_{l \in Leaves} P\left[C_j|\mathbf{X}, A_l\right] P\left[A_l|\mathbf{X}\right] \approx \sum_{l} P\left[C_j|A_l\right] P\left[A_l|\mathbf{X}\right], \tag{3.9}$$

$$P\left[C_j|A_l\right] \approx P^*\left[C_j|A_l\right]. \tag{3.10}$$

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 $P\left[C_j|\mathbf{X}, A_l\right]$ of (3.9) is substituted by the piece-wise constant function $P\left[C_j|A_l\right]$. However, the estimation of $P\left[C_j|A_l\right]$ 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 $P\left[C_j|\mathbf{X}, A_l\right]$ using any predefined model (such as the Gaussian PDF or the uniform PDF) appears easier but is less reliable.

An example is shown in Figure 3.2, 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 $P\left[C_j|X\right]$ 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 worthy. 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

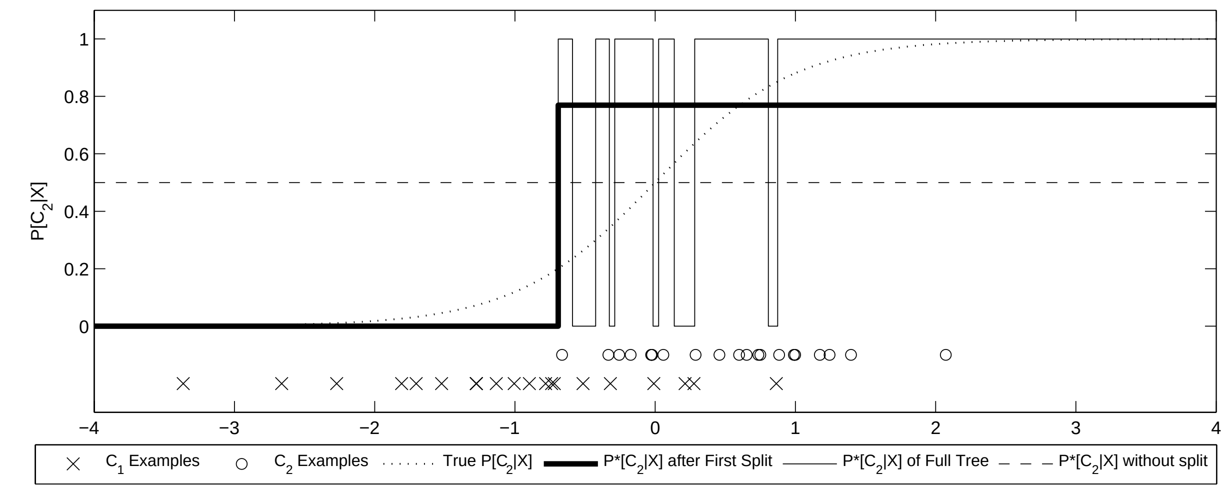


Figure 3.2: Estimated Class Probabilities Obtained by CART for a Gaussian Data Set.

a candidate prune tree, for 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 the next chapter.

3.2 Probability Estimation

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 $\Phi_h (h = 1, 2, \ldots, H)$ 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 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 $n_1, n_2, \ldots, n_H$.

3.2.1 Lidstone's law of succession

Lidstone's law of succession (see [Lid20]) represents posterior estimation, in which p_h is estimated by p_h^*, where

$$p_h^* = E\left[p_h | n_1, \ldots, n_H\right] = \frac{n_h + \lambda_h}{N + \sum_{h=1}^{H} \lambda_h}, \tag{3.11}$$

where λ_h's are predefined non-negative parameters and discussed in subsection 3.2.3.

Lidstone's estimation is widely used in probability estimation, such as in Naive Bayes Classifiers (see [KBS97]). However, most researchers simply compute the expected value according to (3.11). In Theorem 4.3, 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 section 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 unworthy splits. This explains the reason of why, in this dissertation, we also derive expressions for computing standard deviations.

3.2.2 Dirichlet distribution

In Lidstone's estimation, it is assumed (see [Goo67, Goo65]) that the prior distribution of p_h is Dirichlet distribution, denoted by

$$(p_1, \ldots, p_H) \sim Dir(\lambda_1, \lambda_2, \ldots, \lambda_H), \tag{3.12}$$

or

$$f(p_1, \ldots, p_H) = \alpha\delta\left(1 - \sum_{h=1}^{H} p_h\right) \prod_{h=1}^{H} p_h^{\lambda_h - 1} I[p_h \geq 0], \tag{3.13}$$

where

$$\alpha = \frac{\Gamma\left(\sum_{j=1}^{J} \lambda_j\right)}{\prod_{j=1}^{J} \Gamma(\lambda_j)}, \tag{3.14}$$

$\Gamma()$ is the Gamma function, $I[\Phi]$ is the indicator function (1 if Φ is true and 0 otherwise), and $\delta(x)$ is Dirac's delta function (or unit impulse function; see [Zem87]) such that

$$\int_{-\infty}^{\infty} g(x)\delta(x) = g(0), \tag{3.15}$$

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 collective exhaustive. In some works Dirac's delta function in (3.13) is replaced with the indicator function, probably for evaluating the magnitude. Strictly speaking, Dirac's delta function is necessary here because only the degree of freedom of the H random variables is $H - 1$ due to the above constraint (that is, the PDF is non-zero on a $H-1$ dimensional hyperplane only). In order for the integral of the PDF within the H dimensional space to be one, infinite values must be present in the PDF. For example, when $H=1$, p_1 has a deterministic value 1, and its PDF is Dirac's delta function rather than the indicator function. For this reason, in the literature sometimes the joint PDF of $p_1, \ldots, p_{H-1}$ is used to denote the Dirichlet distribution for avoiding Dirac's delta function. Here we use Dirac's delta function to show that $p_1, \ldots, p_H$ are assumed symmetric.

We can derive the expression of the posterior joint PDF of $p_1, \ldots, p_H$ using the prior PDF (3.13) and Bayesian theorem

$$f(p_1, \ldots, p_H | n_1, \ldots, n_H) = \frac{P[n_1, \ldots, n_H | p_1, \ldots, p_H] f(p_1, \ldots, p_H)}{\int P[n_1, \ldots, n_H | p_1, \ldots, p_H] f(p_1, \ldots, p_H) dp_1 \ldots dp_H}, \tag{3.16}$$

as well as the following equation for Bernoulli trials:

$$P[n_1, \ldots, n_H | p_1, \ldots, p_H] = N! \prod_{h=1}^{H} \frac{p_h^{n_h}}{n_h!}. \tag{3.17}$$

Interestingly, the posterior joint PDF turns out to represent another Dirichlet distribution (see [Goo65]):

$$(p_1, \ldots, p_H | n_1, \ldots, n_H) \sim Dir(n_1 + \lambda_1, \ldots, n_H + \lambda_H). \tag{3.18}$$

In order to derive our k-norm estimate, the following equations are necessary (which assume $\lambda_h = \lambda$ regardless of h; see subsection 3.2.3 for details):

$$E\left[p_h^k \middle| n_1, \ldots, n_H\right] = \prod_{m=0}^{k-1} \frac{n_h + \lambda + m}{N + \lambda H + m}, \tag{3.19}$$

$$E\left[(1-p_h)^k \middle| n_1, \ldots, n_H\right] = \prod_{m=0}^{k-1} \frac{N - n_h + \lambda(H-1) + m}{N + \lambda H + m}, \tag{3.20}$$

$$E\left[p_{h_1} p_{h_2} | n_l, \ldots, n_H\right] = \frac{(n_{h_1} + \lambda)\,(n_{h_2} + \lambda + I[h_1 = h_2])}{(N + \lambda H)(N + \lambda H + 1)}. \tag{3.21}$$

Please see Appendix C.1 for the proof of (3.19), (3.20) and (3.21). Equation (3.20) is the basis of our proposed error rate estimation and the corresponding k-norm pruning algorithm. Equations (3.19) and (3.21) are useful for the extensions of our k-norm estimation (see Chapter 7).

3.2.3 Selection and implication of parameters

Usually the same value is assigned to all λ_h's in the prior distribution, according to the *Principle of Indifference*. Therefore, from now on we use the notation λ without subscript. In statistics, a widely used value for λ is 0.5 (Krichevskiy suggests the value 0.50922 in [Kri98]). In our application, the optimal value of λ remains an open problem. Nevertheless, most of our analysis applies to all positive values of λ. In section 6.3, a heuristic is given for our experiments.

When we use a uniform value of λ, we imply that $(p_1, \ldots, p_H)$ are assumed symmetric here, according to their equal parameters in the prior joint PDF. This means that we have no bias towards any class without seeing any training example. An analogy is that before throwing a die, we can treat the six events "the die shows the number h" $(h = 1, 2, \ldots, 6)$ as symmetric, and if so, the events "the die shows the number 6" and "the die does not show the number 6" are not symmetric. In tree classification, we are consistent if we treat the events "**X** has class label j" as symmetric, but we are inconsistent if we treat the events "**X** is correctly classified" versus "**X** is misclassified" as symmetric. For example, under the second definition of symmetric events, if in one tree leaf class 1 is the majority class and classes 2 and 3 are the minority classes, we consider the events "**X** belongs to class 1", and "**X** belongs to class 2 or 3" as symmetric; if, under the same definition, in another leaf of the same tree class 2 is the majority class, and classes 1 and 3 are the minority classes, we consider the events "**X** belongs to class 2", and "**X** belongs to class 1 or 3" as symmetric. Therefore, in our work here, we consider the events "**X** has class label j" as symmetric.

In the existing literature, where Lidstone's law of succession is applied to Machine Learning, some researchers use the terminology "Laplace Law" instead of Lidstone's Law. Laplace's Law was actually proposed before Lidstone's Law, however it represents a special case of Lidstone's law with $\lambda = 1$. Since Lidstone's Law is more general, and more widely used in the statistical literature (see [Ris95]), we refer in this dissertation to the probability estimation approach as Lidstone's approach.

CHAPTER 4
K-NORM ERROR RATE ESTIMATION AND PRUNING

As explained in section 2.4, we apply the Lidstone's law of succession to estimate the error rates. We also rely on the assumptions (Partitioning, Determinancy and Piece-Wise Independence) discussed in subsection 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 chapter 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,

$$E_{\mathbf{X}}[Q] = \int Q f(\mathbf{X}) d\mathbf{X}, \tag{4.1}$$

$$E_{\mathbf{P}}[Q] = \int Q f(\mathbf{P}) d\mathbf{P}, \tag{4.2}$$

$$E[Q] = E_{\mathbf{X}}[E_{\mathbf{P}}[Q]] = E_{\mathbf{P}}[E_{\mathbf{X}}[Q]], \tag{4.3}$$

where $f(\mathbf{X})$ and $f(\mathbf{P})$ are actually posterior PDFs (given the observations $\mathbf{O}$ with the training examples). We omit the condition $\mathbf{O}$ from now on, because all the expected values are conditional.

The (posterior) standard deviations can be computed as follows:

$$\sigma_{\mathbf{P}}[Q] = \sqrt{E_{\mathbf{P}}[Q^2] - E_{\mathbf{P}}[Q]^2}, \tag{4.4}$$

$$\sigma[Q] = \sqrt{E[Q^2] - E[Q]^2}. \tag{4.5}$$

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

$$E_{\mathbf{X}|A_t}[Q] = \int Q f(\mathbf{X}|A_t) d\mathbf{X}. \tag{4.6}$$

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

$$E^t[Q] = E_{\mathbf{P}}\left[E_{\mathbf{X}|A_t}[Q]\right], \tag{4.7}$$

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 4.1 *For any decision node d and any natural number k, under the Partitioning Assumption (the events $\{A_c \, given \, A_d\}$ for any decision node d and $c \in Children(d)$ are mutually exclusive and collectively exhaustive) and the Determinacy Assumption ($P\left[A_c|A_d, \mathbf{X}\right]$ is either 0 or 1), the following expression is valid:*

$$E_{\mathbf{X}|A_d}\left[r_d^k\right] = \sum_{c \in Children(d)} E_{\mathbf{X}|A_c}\left[r_c^k\right] P\left[A_c|A_d\right], \tag{4.8}$$

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

$$r_t = P\left[Err|A_t, \mathbf{X}\right]. \tag{4.9}$$

Proof (The proof of Theorem 1 is provided in Appendix C.2).

Note that $E_{\mathbf{X}|A_d}\left[Q\right]$ represents only the expectation across $\mathbf{X}$, and thus both sides in (4.8) are still random variables.

Corollary 4.2 *Under the Partitioning Assumption (the events $\{A_c \, given \, A_d\}$ for any decision node d and $c \in Children(d)$ are mutually exclusive and collectively exhaustive) and the Determinacy Assumption ($P\left[A_c|A_d, \mathbf{X}\right]$ is either 0 or 1), as well as the assumption that r_c and $P\left[A_c|A_d\right]$ are independent random variables for each child node c of the decision node* *d, the following expression is valid:*

$$E\left[r_d^k\right] = \sum_{c \in Children(d)} E\left[r_c^k\right] E_{\mathbf{P}}\left[P\left[A_c | A_d\right]\right]. \tag{4.10}$$

This corollary can be proven by computing the expected values with respect to $\mathbf{P}$ of both sides of (4.8), according to (4.7). Since $\mathbf{P}$ includes $P[A_c|A_d]$, the other probabilities in $\mathbf{P}$ do not contribute to $E_{\mathbf{P}}\left[P\left[A_c|A_d\right]\right]$ (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). 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_c for all $c \in Children(d)$" (where n_t is the number of training examples covered by node t). Using Lidstone's law of succession, we get

$$E_{\mathbf{P}}\left[P\left[A_c|A_d\right]\right] = \frac{n_c + \eta}{n_d + \eta K_d}, \tag{4.11}$$

where K_d is the number of immediate children of decision node d, and η has an analogous functionality as λ (we use a different notation because λ will be used to denote the parameter in the error rate estimation). For simplicity, from now on, we use $P^*\left[A_c|A_d\right]$ to denote $E_{\mathbf{P}}\left[P\left[A_c|A_d\right]\right]$.

Using the corollary, the standard deviation of the error rate can be computed as follows.

$$E\left[r_d\right] = \sum_{c \in Children(d)} E\left[r_c\right] P^*\left[A_c|A_d\right], \tag{4.12}$$

$$E\left[r_d^2\right] = \sum_{c \in Children(d)} E\left[r_c^2\right] P^*\left[A_c|A_d\right], \tag{4.13}$$

$$\sigma\left[r_d\right] = \sqrt{E\left[r_d^2\right] - E\left[r_d\right]^2}. \tag{4.14}$$

It is important to note that generally,

$$\sigma\left[r_d\right] \neq \sum_{c \in Children(d)} \sigma\left[r_c\right] P^*\left[A_c|A_d\right]. \tag{4.15}$$

Theorem 4.1 makes our error rate estimation stronger than most other approaches, such as EBP (see section 2.3). It can also be generalized to other risk definitions (see Chapter 7).

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 (3.5), before splitting,

$$\begin{aligned}
E\left[r_d\right] &= E_{\mathbf{P}}\left[E_{\mathbf{X}}\left[P\left[Err|A_d,\mathbf{X}\right]\right]\right] = E_{\mathbf{P}}\left[P\left[Err|A_d\right]\right] \\
&= E_{\mathbf{P}}\left[1 - P[C_1|A_d]\right] = \frac{1+0.5}{99+0.5\times 2} = 0.015000, \qquad (4.16) \\
E\left[r_d^2\right] &= E_{\mathbf{P}}\left[E_{\mathbf{X}}\left[P\left[Err|A_d,\mathbf{X}\right]^2\right]\right] \approx E_{\mathbf{P}}\left[P\left[Err|A_d\right]^2\right] \\
&= E_{\mathbf{P}}\left[(1 - P[C_1|A_d])^2\right] = \frac{(1+0.5)(1+0.5+1)}{(99+0.5\times 2)(99+0.5\times 2+1)} \\
&= 0.00037129, \qquad (4.17) \\
\sigma\left[r_d\right] &= \sqrt{E\left[r_d^2\right] - E\left[r_d\right]^2} \approx 0.012095. \qquad (4.18)
\end{aligned}$$

After splitting, for the left child c_1,

$$E\left[r_{c_1}\right] = \frac{0+0.5}{98+0.5\times 2} = 0.0050505, \qquad (4.19)$$

$$E\left[r_{c_1}^2\right] \approx \frac{(0+0.5)(0+0.5+1)}{(98+0.5\times 2)(98+0.5\times 2+1)} = 0.00075758, \qquad (4.20)$$

$$P^*\left[A_{c_1}|A_d\right] = \frac{98+0.5}{99+0.5\times 2} = 0.98500. \qquad (4.21)$$

After splitting, for the right child c_2,

$$E\left[r_{c_2}\right] = \frac{0+0.5}{1+0.5\times 2} = 0.25000, \qquad (4.22)$$

$$E\left[r_{c_2}^2\right] \approx \frac{(0+0.5)(0+0.5+1)}{(1+0.5\times 2)(1+0.5\times 2+1)} = 0.12500, \qquad (4.23)$$

$$P^*\left[A_{c_2}|A_d\right] = \frac{1+0.5}{99+0.5\times 2} = 0.015000. \qquad (4.24)$$

For the sub-tree,

$$E\left[r_d\right] = 0.0050505 \times 0.985 + 0.25 \times 0.015 = 0.0087247, \tag{4.25}$$

$$E\left[r_d^2\right] \approx 0.00075758 \times 0.985 + 0.125 \times 0.015 = 0.0019496, \tag{4.26}$$

$$\sigma\left[r_d\right] \approx \sqrt{0.0019496 - (0.0087247)^2} = 0.04328. \tag{4.27}$$

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. Therefore, we do not expect that this split will improve the generalization.

4.4 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, a statement that is validated by the standard deviations, computed in the previous example. Therefore, it is reasonable to consider combining the expected value and the standard deviation to estimate the error rate (e.g., by using $E[r^2] = E[r]^2 + \sigma[r]^2$ (or $\sqrt{E\left[r^2\right]}$) as our error rate estimate at a tree leaf). In general, we use k-norm estimation, which estimates $\|r\|_k = \sqrt[k]{E\left[|r|^k\right]}$ (which is equal to $\sqrt[k]{E\left[r^k\right]}$ because in this dissertation $r \geq 0$). Our choice is supported by the desirable properties (see Chapter 5) of the k-norm pruning algorithm based on the k-norm estimation.

Apparently, the error rate of the tree is equal to $r = r_{root}$. Under the *Partitioning Assumption* and the *Determinacy Assumption* of subsection 3.1.1, for a decision node d,

$$E\left[r_d^k\right] = \sum_{c \in Children(d)} E\left[r_c^k\right] P^*\left[A_c | A_d\right]. \tag{4.28}$$

For a leaf l, define J as the number of classes. Under the *Piece-wise Independence Assumption* (3.5),

$$\begin{aligned} E\left[r_l^k\right] &= E\left[(1 - P[C_{Label(l)}|A_l, \mathbf{X}])^k\right] \\ &\approx E\left[(1 - P[C_{Label(l)}|A_l])^k\right] \\ &= \prod_{i=0}^{k-1} \frac{n_l - n_{j,Label(l)} + (J-1)\lambda + i}{n_l + J\lambda + i} \\ &= \prod_{i=0}^{k-1} \frac{n_l - \max\limits_j n_{j,l} + (J-1)\lambda + i}{n_l + J\lambda + i}. \end{aligned} \tag{4.29}$$

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 below:

The 1-norm error rate is the expected value of the error rate. The 2-norm $\|r\|_2 = \sqrt{E\left[r\right]^2 + \sigma\left[r\right]^2}$ includes the expected value and the standard deviation. This is similar to the "Bias-Variance" decomposition in [BFO84] (page 88), while the definition of the variance

```
Algorithm: $R = \mathrm{PruneTree}(t)$
input  : a tree rooted at node $t$
output: the optimal pruned tree (modified from the input),
        and its $k$-th moment error rate $R$ (returned value of this function)
Compute $R_{leaf} = \prod_{m=0}^{k-1} \frac{n_t - n_{Label(t),t} + \lambda(J-1) + m}{n_t + \lambda J + m}$;
if $t$ is a decision node then
    Compute $R_{tree} = \sum_{c \in Children(t)} \frac{n_c + \eta}{n_t + \eta K_t}$ PruneTree ($c$);
    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 $R_{leaf}$;
```

Figure 4.1: Pseudo Code of Algorithm PruneTree(t)

in our case is easier to understand, and can be applied to multi-class problems and to any risk definition. For higher k values, the k-norm also includes the skewness (for $k \geq 3$) and/or the kurtosis (for $k \geq 4$) is also included. The behavior of the k-norm pruning algorithm is discussed in Chapter 5. We do not recommend $k = 1$ for the k-norm estimation because it is too optimistic as an error rate estimate (see also [EMS97]), and theoretically demonstrated by the following theorem.

Theorem 4.3 *Under the Partitioning Assumption (the events $\{A_c given A_d\}$ for any decision node d and $c \in Children(d)$ are mutually exclusive and collectively exhaustive) and the Determinacy Assumption ($P[A_c|A_d, \mathbf{X}]$ is either 0 or 1), if* $0 \leq \eta \leq \lambda J$, $\lambda \leq 1/(K_t-1)(J-1)$ *at a node t, 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.3).

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, as long as they are effective.

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 $E_{\mathbf{P}}\left[r^k\right]$ where $r = P\left[Err|\mathbf{X}\right]$. We also apply the k-norm estimation here. Under the assumptions of subsection 3.1.1, only one leaf $l_{\mathbf{X}}$ receives $\mathbf{X}$. Therefore,

$$\begin{aligned} E_{\mathbf{P}}\left[r^k\right] &= E_{\mathbf{P}}\left[\left(1 - P\left[C_{Label^*(\mathbf{X})}|\mathbf{X}, A_{l_{\mathbf{X}}}\right]\right)^k\right] \\ &\approx E_{\mathbf{P}}\left[\left(1 - P\left[C_{Label(l_{\mathbf{X}})}|A_{l_{\mathbf{X}}}\right]\right)^k\right] \\ &= \prod_{i=0}^{k-1} \frac{n_{l_{\mathbf{X}}} - \max\limits_j n_{j,l_{\mathbf{X}}} + \lambda(J-1) + i}{n_{l_{\mathbf{X}}} + \lambda J + i}. \end{aligned} \tag{4.30}$$

In practice, there are two reasonable ways to output the estimated error rate: 1) Output the 2-norm error rate $\|r\|_2 = \sqrt{E_{\mathbf{P}}\left[r^2\right]}$ only, or 2) Output $E_{\mathbf{P}}\left[r\right]$ and $\sigma_{\mathbf{P}}\left[r\right] = \sqrt{E_{\mathbf{P}}\left[r^2\right] - E_{\mathbf{P}}\left[r\right]^2}$, the latter 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.

CHAPTER 5
PROPERTIES OF K-NORM ESTIMATION/PRUNING

The k-norm pruning algorithm has a number of properties that are similar to CCP properties of CART. A case study in Appendix B shows some of these similarities between k-norm pruning algorithm and CART's Minimal Cost-Complexity Pruning (CCP) algorithm. This example serves as both the motivation and the application of our analysis and theorems on the properties of the k-norm pruning algorithm provided in this chapter. More importantly, since CCP's properties have undoubtable practical significance (see the context for details), we successfully demonstrate the same virtues of the k-norm pruning algorithm by showing that the k-norm pruning algorithm has analogous properties. Furthermore, we show that the k-norm pruning algorithm has additional desirable properties that CCP does not possess.

5.1 Similarities Between k-norm pruning algorithm and CCP

5.1.1 Property 1: each branch in the optimal pruned tree is also optimal

Even for binary trees, the number of possible pruned trees is typically $\Theta(1.5^L)$ (see [BFO84], page 284), where L is the number of leaves in the fully grown tree. Therefore, it is impractical to choose the best pruned tree by exhaustive enumeration. A general improvement has been proposed in [BB94], where dynamic programming is used to reduce the computational complexity to $\Theta(L^2)$. For both CCP and our k-norm pruning algorithm, Property 1 allows us to minimize the computational complexity to $\Theta(L)$, if the parameters (α for CCP and k for k-norm) are fixed: only one tree traversal is necessary to find the optimal pruned tree, and the chosen tree is guaranteed optimal among all possible pruned trees although we do not visit all the $\Theta(1.5^L)$ pruned trees.

For example, CART grows a tree with 1059 leaves on the *abalone* database (see [NHB98]), if all stopping criteria are disabled (that is, the growing process continues until each leaf is pure or no split can be found). There are 1.67×10^{150} possible pruned trees according to the equation given in [BFO84], page 284. For CCP and k-norm pruning algorithm, only 2117 nodes need to be visited, if the parameters α or k are given, respectively.

CCP of CART applies a cost heuristic (shown below) to evaluate the pruned trees:

$$r_t = R_t + \alpha L_t, \tag{5.1}$$

where R_t is the number of training misclassifications made by the leaves of the sub-tree rooted at t, L_t is the number of these leaves, and α is a parameter. It is easy to see that the cost of a sub-tree is the sum of the costs of its branches. That is, for a decision node d,

$$r_d = \sum_{c \in Children(d)} r_c. \tag{5.2}$$

In the optimal pruned tree, each branch must also be optimally pruned. If for one branch T in the optimal pruned tree T_0 we can find a pruned branch T' with less cost, we can replace T with T' and the total cost of the pruned tree can be reduced, violating our assumption that T_0 has the smallest cost. Therefore, to find the optimal pruned tree, we can first find the optimal branches first, resulting in a recursive, divide-and-conquer CCP algorithm, which can finish in only one tree traversal.

For our k-norm pruning algorithm, the optimal pruned tree is the one minimizing the k-norm error rate, or equivalently, the k-th moment of the error rate. According to Theorem 4.1, the k-th moment of the error rate of a sub-tree is also a weighted sum of those of its branches. The weights do not depend on how the branches are pruned. Therefore, the above analysis of CCP also applies to our k-norm pruning algorithm.

When the parameter values (α or k) vary, different optimal pruned trees will be found. Therefore, finding the final tree may require more computations, including the search of all optimal pruned trees corresponding to various parameter values, as well as the evaluation of each tree. We have shown that this step is not necessary for k-norm pruning because we can fix $k = 2$, and obtain good results. However, it is worth considering how to speed-up this step in order to show that our algorithm is still practical if k is allowed to take different values.

5.1.2 Property 2: higher parameter values prune more nodes

To choose the best pruned tree by varying the parameter α or k within M values, the naive way is to repeat finding and evaluating the optimal pruned tree for each value, resulting in total time complexity $\Theta(ML) + \Theta(MV)$, where V is the time complexity for evaluation. The evaluation is usually accomplished by cross-validation or the validation of the trees on a separate set of data (in either way, V is at least linear to N, the size of the data for validation, and it is typically $\Theta(N \log L)$). For CCP and k-norm, no preset value of M can guarantee that all distinct pruned trees will be found. Fortunately, Property 2 implies that we can find out all distinct pruned trees without relying on M, as explained below:

- If the parameter is raised, the optimal pruned tree is a further pruned version of the previous one, meaning that all the optimal pruned trees can be sorted in a pruning sequence, and each member tree of this pruning sequence corresponds to parameter values organized in ascending order. The pruning sequence can be found by the following: 1) for each decision node d, computing the parameter value at which the sub-tree rooted at d is pruned, and 2) repeat finding the decision nodes with the smallest parameter value, pruning the corresponding sub-tree, and updating the parameter values associated with the ancestor nodes of the pruned sub-tree. The time complexity of finding all the pruned trees is $\Theta(L^2)$ in the worst case (when the fully grown tree is a linear tree and each time only one decision node is pruned) and $\Theta(L \log L)$ in typical cases (when the fully grown tree is a balanced tree and each time only one decision node is pruned).

- Since each pruned tree in the pruning sequence is a further pruned version of the previous tree, we do not have to pass the validation examples to all pruned trees. Instead, if we pass the examples only to the fully grown tree, record the class distribution in each decision node, compute the error rate of the fully grown tree, we get the performance of each pruned tree by the following process. For each pruned tree, the number of errors it makes is equal to that of its previous tree, plus the difference caused by the pruned branches from the previous tree, which can be computed by one traversal of the pruned branches. The evaluation of the fully grown tree has time complexity $\Theta(V)$, and the computation of the error rate of all the pruned trees add up to $\Theta(L)$, because from the fully grown tree to the last pruned tree, at most $L-1$ nodes are pruned, and thus the time complexity of the traversal of all the pruned branches adds up to $\Theta(L)$.

Therefore, Property 2 makes the selection of the final tree affordable, typically with time complexity $\Theta(L \log L) + \Theta(N \log L)$. Furthermore, the parameter values corresponding to each distinct pruned tree can be computed as a by-product, rather than being preset by "shooting in the dark".

For CCP, this property is proven in [BFO84], page 70. Here we prove that the k-norm pruning algorithm has exactly the same property by first stating two lemmas that are needed for the proof of a theorem (Theorem 5.3) and a corollary that proves Property 2.

Lemma 5.1 *If two leaves have the same k-norm error rate, the smaller leaf (the leaf receiving fewer training instances) has higher $(k+1)$-norm error rate.*

This lemma indicates that small leaves are penalized more when k increases. This property is intuitive because we know that the probabilities estimated using fewer observations are less reliable.

Lemma 5.2 *If the weighted average of the k-th moment error rates of a set of leaves is equal to that of another leaf, and if all leaves cover an equal number of training examples, then the weighted average (with the same weights as before) of the $(k+1)$-th moment error rate of the same set of leaves is no less than the $(k+1)$-th moment of the other leaf. The equality holds if and only if all leaves have the same number of training misclassifications.*

This lemma implies that the diversity (different error rates) among leaves will be punished more with higher k value. For example, it is not difficult to see that both the sub-trees in Figure 5.1 have a 1-norm error rate of 12% when $\lambda = 1$, regardless of η, and all leaves cover 48 examples. When k is increased to 2, the k-norm error rate of the left sub-tree is increased to 16.2%, while the k-norm error rate of the right sub-tree is increased slightly to 12.8%.

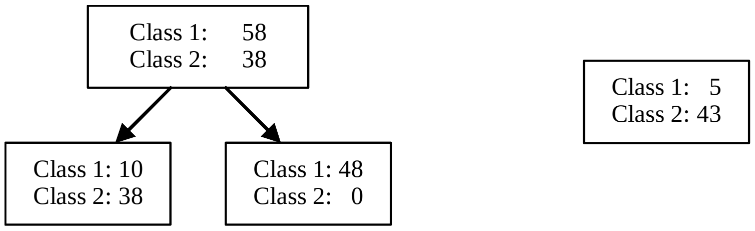


Figure 5.1: Sub-trees with Same 1-Norm Error But Different 2-Norm Errors

Theorem 5.3 *If a non-leaf sub-tree increases or maintains the k-norm error rate compared to its root, it increases the $(k + 1)$-norm error rate.*

Corollary 5.4 *Let T_k denote the optimal pruned sub-tree of T that minimizes the k-norm error rate, T_{k+1} is a pruned version of T_k for any $k > 0$.*

An example of the pruning sequence is shown in Figure B.1 with *iris* database. Another example pruning sequence containing many more pruned trees is shown in Figure 5.2. In this example we grow a tree with CART and the database *abalone*, and prune the tree with various values of k.

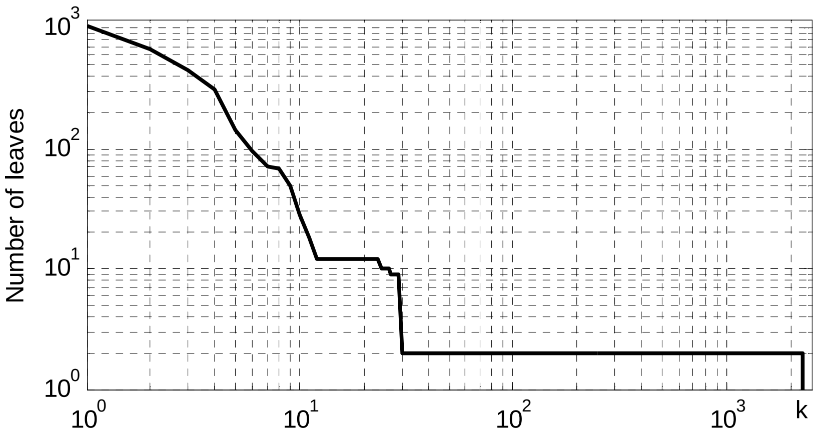


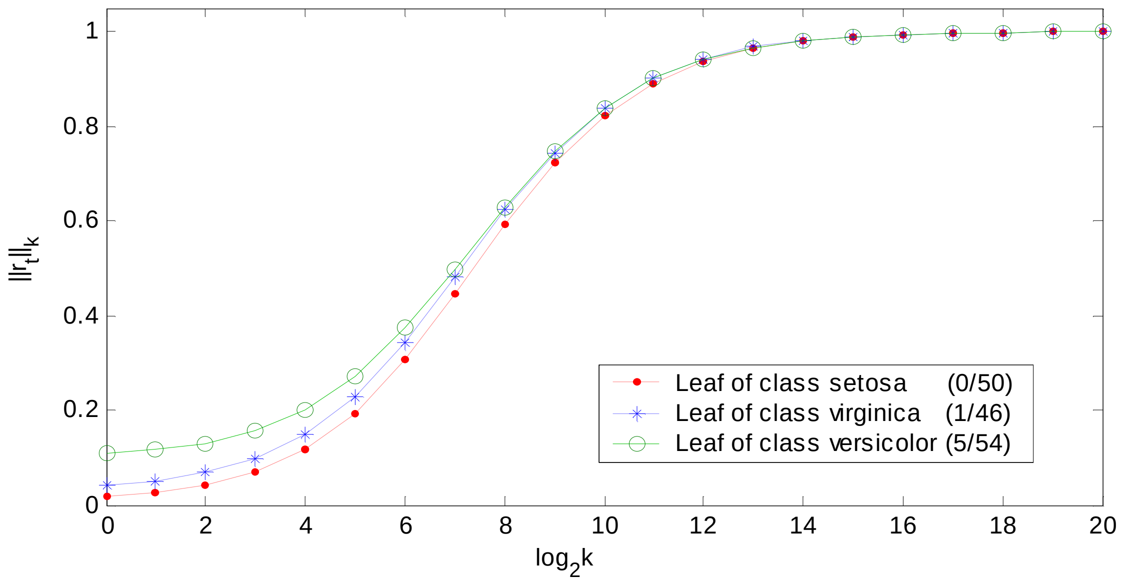
Figure 5.2: Size of Pruned Tree versus k on *Abalone* Database

5.1.3 Property 3: higher parameter values mean more pessimism

It is easy to see that when α in (5.1) increases, the cost is higher for any node. For the k-norm pruning algorithm, we have the same property:

Theorem 5.5 *For any sub-tree, its k-norm error rate is a strictly increasing function of k.*

This theorem indicates that we can control how pessimistic the estimation is by tuning the parameter. For an illustration of the increase of the k-norm error rates, we choose the tree shown in Figure B.1(d) and Figure B.2, and plot the k-norm error rates of its leaves with respect to k in Figure 5.3.



Note: "0/50" in the legend means 0 misclassification out of 50 training examples in the leaf.

Figure 5.3: k-Norm Leaf Error Rates of the Tree in Figure B.1(d)

5.1.4 Property 4: any optimal pruned tree removes ineffective splits

For any $\alpha \geq 0$, CCP removes all ineffective splits, which are defined as those that do not reduce the training misclassifications. This property is apparently desirable because ineffective splits make no difference in performance but increase the model complexity (here we assume that in a leaf, if two classes have the same number of examples, the first class is chosen). For the k-norm pruning algorithm, we found the same property:

Theorem 5.6 *At a node t, if $0 \leq \eta \leq \lambda J$, any ineffective split never decreases the 1-norm error rate; it maintains the 1-norm error rate if and only if all the classes are equally distributed in each its leaf. A split is called effective if it reduces the training misclassifications and ineffective otherwise.*

Corollary 5.7 *If $0 \leq \eta \leq \lambda J$, any ineffective split increases the k-norm error rate for any $k \geq 1$ (when $k = 1$, it can be seen from Theorem 5.6; when $k > 1$, it can be induced with Theorem 5.3).*

Ineffective splits are not commonly seen, but they are possible to occur when the training set contains conflicting examples (two or more examples with the same attributes but different classes). See Appendix B for an example of an ineffective split.

5.2 Unique Characteristics of k-Norm Pruning Algorithm

In the previous section we have shown that the k-norm pruning algorithm has similar desirable properties as CCP, with significant practical values. Now we list some other desirable properties that only the k-norm pruning algorithm possesses.

- **Property 5:** The k-norm algorithm may not need validation for finding the pruning sequence at all. It can be shown that fixing $k = 2$ yields sufficiently good results, with negligible time complexity (see Chapter 6). For CCP, there is no default value of α, and therefore validation is necessary, resulting in higher time complexity (see Figure 6.2 for the difference on large databases).

- **Property 6:** The k-norm algorithm yields an error rate estimate with clear theoretical interpretation. Our k-norm error rate estimate is always between 0 and 1, increasing smoothly and approaching 1 as k increases, as shown by Theorem 5.8. This means that our estimate is meaningful for all natural numbers k; if we are infinitely pessimistic (by setting $k = \infty$), we postulate that the prediction of every node is definitely wrong, and as a result, no split is worthy.

  Furthermore, we can also fix $k = 2$, and the 2-norm estimate can be treated as an error rate. For CCP, only a range of α values can produce meaningful predicted cost in [0,1], and this range of α depends on the tree, resulting in the following difficulties. If we preset $\alpha = 0.1$, the cost of a tree with 100 leaves has a cost 10 which cannot be treated as an error rate.

Theorem 5.8 *For any sub-tree, the limit of its k-norm error rate is unity as k approaches ∞.*

Please refer to Figure 5.3 for an example.

In contrast, CCP's cost estimate in (5.1) does not have an obvious correspondence to the generalization error rate, although it is based on the regularization approach that has been used extensively, and it can approximate the error rate with proper values of α found by cross-validation. Furthermore, only a range of α value can produce meaningful predicted cost in [0,1], and this range of α depends on the tree, which means that we cannot preset α before a tree is given for error rate estimation. For example, suppose a real-time system is required to estimate the error rate of a tree without any data set (which means cross-validation is not applicable). When the tree is given , α is somehow chosen to a value and an estimation is made; the α may be out of range for all the subsequent input trees. If we decide to reduce α at some point, the previous estimation results are not comparable to future results.

- **Property 7:** Sometimes only very large values of k can over-prune a tree (see Figure B.1(e) and Figure B.1(f)), which provides a protection from over-pruning.

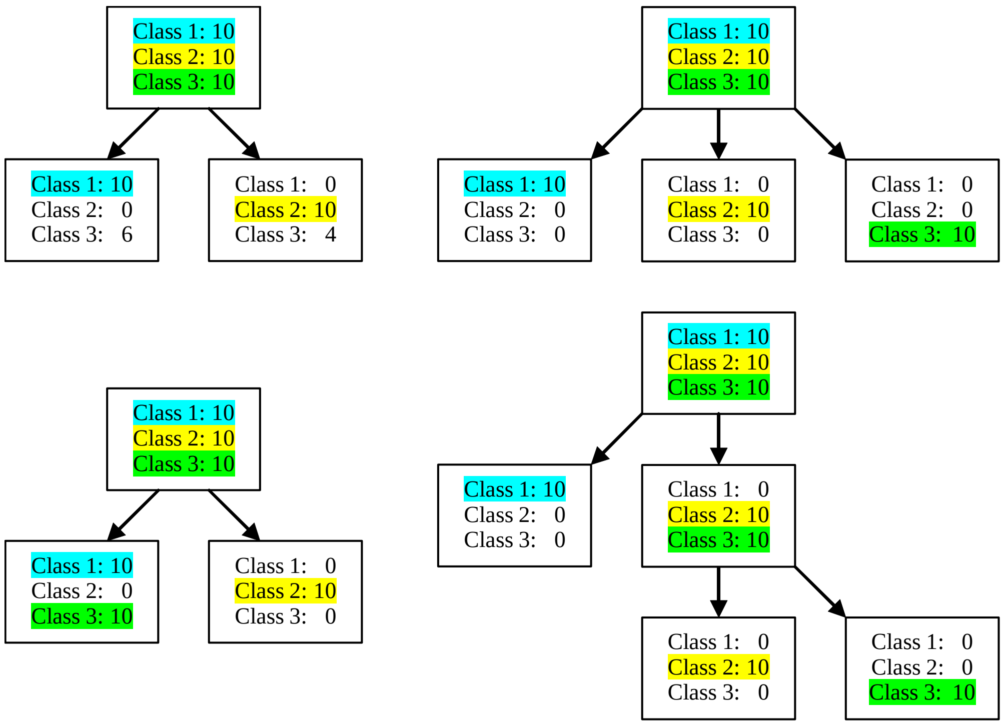
In particular, some splits can survive under any finite value of k. For CCP, any split will be pruned by a finite value of α (less than 1), even if they are beneficial splits. For example, the last tree in Theorem 5.9 represents the optimal solution (assuming it has no ancestor node). This tree will not be pruned for any $k < \infty$ by k-norm pruning, but it will be pruned to a single leaf by CCP when $\alpha = 1/3$.

The importance of this property is that tree splits that with k-norm pruning can survive pruning for a large number of k values can be safely be thought of as very useful for the tree model. Furthermore, the following theorem states that a split that survives pruning under any finite k value has to satisfy very strict conditions (see Theorem 5.9), and as a result it rarely happens.

Theorem 5.9 *A necessary and sufficient condition for a subtree rooted at d to reduce the k-norm error rate for all* $k \in [1, \infty)$ *is that* $\max_j n_{j,d} = \max_j n_{j,c}$ *for all its leaves c.*

This condition is very strict because: 1) node d must have more than one majority classes (so that at least one non-label class will not be treated as noise even when extremely pessimistic), and 3) each leaf under d must cover *all* examples of at least one majority class (which means the splits are perfect). Furthermore, this condition is not a sufficient condition, but a necessary one, for the subtree to remain in the optimal pruned tree for all $k < \infty$, since one of the ancestors of the subtree might be pruned.

Theorem 5.9 is further explained by referring to Figure 5.4. In the top-left tree in Figure 5.4, if in a decision node, 10 examples belong to class 1, 10 examples belong to class 2, and less than 10 examples belong to each other class, and all the 10 examples of class 1 are sent to its left child, while all the 10 examples of class 2 are sent to its right child, this split is *locally* beneficial under any k-norm criterion, because it perfectly separates the two



Class 1: 10 Class 2: 10 Class 3: 10
Class 1: 10 Class 2: 0 Class 3: 6
Class 1: 0 Class 2: 10 Class 3: 4
Class 1: 10 Class 2: 10 Class 3: 10
Class 1: 10 Class 2: 0 Class 3: 0
Class 1: 0 Class 2: 10 Class 3: 0
Class 1: 0 Class 2: 0 Class 3: 10
Class 1: 10 Class 2: 10 Class 3: 10
Class 1: 10 Class 2: 0 Class 3: 10
Class 1: 0 Class 2: 10 Class 3: 0
Class 1: 10 Class 2: 10 Class 3: 10
Class 1: 10 Class 2: 0 Class 3: 0
Class 1: 0 Class 2: 10 Class 3: 10
Class 1: 0 Class 2: 10 Class 3: 0
Class 1: 0 Class 2: 0 Class 3: 10

Note: Majority classes are highlighted.

Figure 5.4: Pruned Trees that Survive under Any Finite k

majority classes. For this reason, the smallest optimal pruned tree at $k < \infty$ can never be over-sized, because the number of leaves is at most equal to the number of majority classes. This is also true for the other trees shown in Figure 5.4, because every leaf also cover all the examples of at least one majority class.

Of course, if the examples of both majority classes are noise, one the ancestors of this decision node is likely to be pruned, and this split no longer exists in the final tree (for this reason we say it is locally beneficial). On the other hand, if only 9 examples belong to class 2, this split cannot survive eventually because when we are pessimistic enough, we can treat the 9 examples of class 2 as noise even if the 10 examples of class 1 are not.

CHAPTER 6
EXPERIMENTS

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

6.1 Procedure

We show three metrics of the compared algorithms: the elapsed time in pruning (computational complexity of pruning), the accuracy of the pruned tree, and the tree size. 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 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 cross-validation 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 $i \mod 20, (i+1) \mod 20, \ldots, (i+m-1) \mod 20$ 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%, $m = 10$). 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 below:

```
foreach Database D do
    Shuffle the examples in $D$;
    Partition $D$ into 20 subsets $D_0, D_1, \ldots, D_{19}$;
    for $m$ = 1, 10 do
        for $i$ = 0 to 19 do
            Set $TrainingSet = \bigcup_{j=0}^{m-1} D_{(i+j) \mod 20}$;
            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
```

Figure 6.1: Pseudo Code of Experimental Procedure

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 by 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 2.59GHz, 1.5GB Memory, and Windows XP). No file I/O, screen display, or keyboard input, but only number crunching is involved during timing. We used a high resolution timer whose result is accurate to one microsecond.

6.2 Databases

To ensure that our experimental results are statistically significant, we require that the databases must have at least 2000 examples. Furthermore, our algorithm does not handle missing values at this time according to the *Determinacy Assumption* (see the discussion in subsubsection 8.2.2.2), and therefore we choose only the databases without missing values. The statistics of the databases that we experimented with are listed in Table 6.1.

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 (see [NHB98]), according to our criteria mentioned in the beginning of this section.

Table 6.1: Statistics of Databases Used in 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

6.3 Parameters

In the experiments, we choose $k = 2$ because we consider it necessary to include the standard deviation of the error rate in our estimation, but we do not see the benefits of taking the skewness into account in our experiments. 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 the overlap amount of the database. 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 (4.29). For highly-overlapped 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. 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 B.

For CCP and EBP, the default parameters were used.

6.4 Results

6.4.1 Accuracy

The accuracies of the tree pruned by each algorithm are listed in Table 6.2. 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

Table 6.2: **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 lose			5 wins, 1 lose		
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 win, 0 lose			7 wins, 0 lose		

- R is the training ratio (%).
- Mean and Std are the mean and the standard deviation, respectively, of the accuracy (%) among 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.

the 2-norm algorithm. The cause is that when the size of the training set is very small, 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, 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 6.3).

6.4.2 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 Table 6.3. 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 6.3) that C4.5 always generates larger trees than the other two.

Table 6.3: Comparison of Tree Size (Leaves) 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 loses			14 wins, 0 lose		
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 loses			14 wins, 0 lose		

- R is the training ratio (%).
- Mean and Std are the mean and the standard deviation, respectively, of the tree size (number of leaves) among 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 Sig is at least 95. 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.

6.4.3 Speed

The mean elapsed time (in seconds) of the three pruning algorithms is shown in Table 6.4. 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 6.4: Comparison of Elapsed Time (Seconds) 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.

Table 6.4 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 access any training example and it finishes the tree evaluation within one traversal of the tree. Although most of the times reported in Table 6.4 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 is better in scalability than CCP and EBP. The Inc Columns in Table 6.4 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 (see [Sto78] and [BFO84]). 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 $\Theta(N^2 \log N)$ is justified in [DHS00], page 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 $\Omega(N)$.

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.

6.5 Additional Experiments

6.5.1 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.

The speed difference shown in Figure 6.2 between our algorithm and CCP and EBP is apparent. The time required by our algorithm is linear with respect to the size of these databases (because for these databases, the size of the fully grown tree turns out to be linear with respect to 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 (e.g., $\log(y) = 2\log(x)$) represents a quadratic function in linear scale (e.g., $y = x^2$). 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

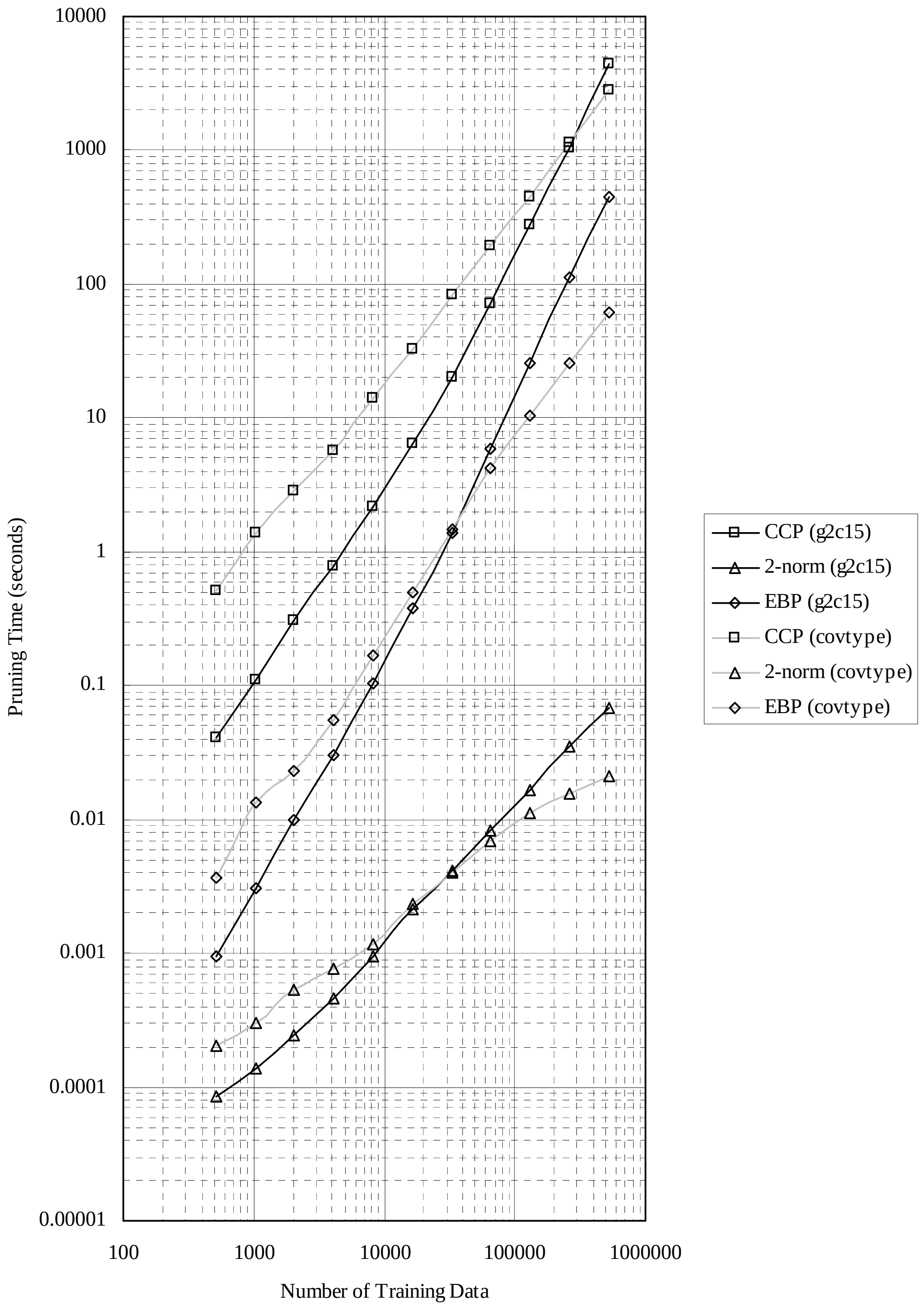


Figure 6.2: Pruning Time (Seconds) of CCP, 2-Norm and EBP on Large Databases (Gaussian and Cover Type) as Training Set Size Increases

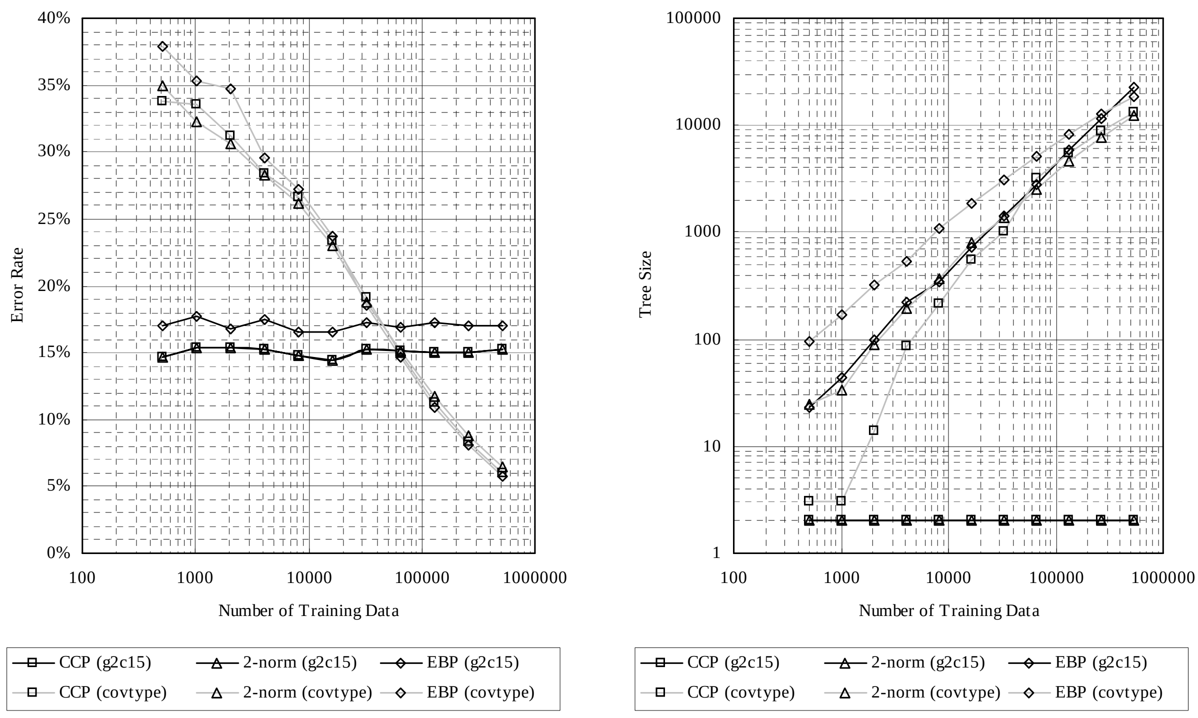


Figure 6.3: Error Rate (%) and Tree Size (Leaves) of CCP, 2-Norm and EBP on Large Databases (Gaussian and Cover Type) as Training Set Size Increases

requires 0.021 seconds, compared to 61.3 seconds and 2848 seconds required by EBP and CCP, respectively.

As shown in Figure 6.3, 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.

6.5.2 Other databases

To compare our algorithm with other pruning algorithms (beyond CCP and EBP) we used the results included in [EMS97]. In particular, we ran the databases in [EMS97] using the same protocol used there (70% of the data were used for training; experiments were repeated 25 times). At this stage, we do not consider missing values, and as a result only 4 of the databases considered in [EMS97] were chosen.

The results in Table 6.5 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 6.5) is close to that presented in [EMS97] (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 given in this dissertation to the performance of the other pruning algorithms given in [EMS97].

Table 6.5: Error Rate (%) and Tree Size (Leaves) 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 medium performances, respectively, of the pruning algorithms reported in [EMS97]. The error rate comparison is shown in the upper half of the table, and the size comparison is shown in the lower half.

CHAPTER 7
EXTENSIONS OF K-NORM ESTIMATION/PRUNING

In this chapter, we show that our k-norm estimation can be generalized to two other tree risk definitions: the misclassification cost (given a loss matrix) and the error of the class probabilities. To do so, we first state a theorem generalized from Theorem 4.1:

Theorem 7.1 *If the events $\{A_c|A_d\}_{c \in Children(d)}$ are mutually exclusive and collectively exhaustive and $P[A_c|A_d, \mathbf{X}]$ is either zero or one for any $\mathbf{X}$, then for any decision node d, any natural number k, any constants a_h, and any events Φ_h,*

$$E_{\mathbf{X}|A_d}\left[\left(\sum_h a_h P[\Phi_h|A_d, \mathbf{X}]\right)^k\right] = \sum_{c \in Children(d)} E_{\mathbf{X}|A_c}\left[\left(\sum_h a_h P[\Phi_h|A_c, \mathbf{X}]\right)^k\right] P[A_c|A_d], \tag{7.1}$$

where $P[\Phi_h|A_d, \mathbf{X}]$ can be the actual or the estimated probability, as long as $P[\Phi_h|A_d, \mathbf{X}] = \sum_{c \in Children(d)} P[\Phi_h|A_c, \mathbf{X}] P[A_c|A_d, \mathbf{X}]$.

The proof of this theorem is analogous to that of Theorem 4.1 (see Appendix C.2). Theorem 7.1 states that if any risk can be defined as a linear summation of a set of probabilities, the recursive evaluation of the k-norm estimate is still valid, and so is the k-pruning algorithm shown in Figure 4.1 (with the minor change in the evaluation of leaves). In the next two sections, we show that the proper definitions of a_h can lead to more meaningful results.

7.1 Misclassification Costs Given a Loss Matrix

In this section, we focus on the cases where 1) the misclassification cost is not uniform as may well be the case in practice, and 2) the overall cost of a tree and that of each its prediction need to be estimated, a desirable requirement in a number of practical situations. For example, misclassifying a healthy patient as ill usually has a different cost as misclassifying an ill patient as healthy. Furthermore, it would be more helpful to provide not only the diagnosis but also the associated estimated cost of such a diagnosis. Given any algorithm that handles these cases, one can design a pruning algorithm that selects the pruned tree to minimize the estimated cost. Unfortunately, to the best of our knowledge the only widely cited pruning algorithms that handle both the above cases in the literature are CCP (see [BFO84], page 66), Laplace-based pruning (which can be treated as MEP [NB86, CB91] for misclassification costs, see [BKK98]), and their hybrids. In [BKK98], it has been illustrated that the optimal tree depends on the loss matrix, and thus a classifier optimized by minimizing the error rate (in which case the misclassification cost is treated as uniform) is not necessarily optimal in the sense of misclassification costs. It has also been concluded by experimentation that Laplace-based pruning has the best performance but the hybrid CCP with Laplace correction and the no-pruning version with Laplace correction also produce good results. Since MEP corresponds to the special case of $k = 1$, we provide below a general k-norm estimate of misclassification costs.

7.1.1 Estimation for sub-trees

To estimate the misclassification cost, a $J \times J$ loss matrix $\mathbf{R} = \{R_{i,j}\}$ must be given, where $R_{i,j}$ is the misclassification cost when the predicted label is i while the actual one is j (regardless of $\mathbf{X}$). We can express the misclassification cost of the sub-tree rooted at d as:

$$r_d = \sum_{i=1}^{J} \sum_{j=1}^{J} R_{i,j} P\left[C_i^*, C_j | \mathbf{X}, A_d\right], \tag{7.2}$$

where C_i^* is the event that $\mathbf{X}$ is classified as belonging to class i,

$$P\left[C_i^*, C_j | \mathbf{X}, A_d\right] = P\left[C_j | \mathbf{X}, A_d\right] I[i = \mathit{Label}^*(\mathbf{X})], \tag{7.3}$$

and $\mathit{Label}^*(\mathbf{X})$ is the predicted label of $\mathbf{X}$, namely $\mathit{Label}(l_{\mathbf{X}})$ where $l_{\mathbf{X}}$ is the leaf that covers $\mathbf{X}$.

Using Theorem 7.1 and setting $\Phi_{i,j} = (C_i^*, C_j), a_{i,j} = R_{i,j}$, we can prove that (4.10) is also true for misclassification costs, that is,

$$E\left[r_d^k\right] = \sum_{c \in Children(d)} E\left[r_c^k\right] E_{\mathbf{P}}\left[P\left[A_c | A_d\right]\right]. \tag{7.4}$$

7.1.2 Estimation for leaves

For a leaf l, the misclassification cost r_l given an input $\mathbf{X}$ is:

$$r_l = \sum_{j=1}^{J} R_{Label(l),j} P\left[C_j|\mathbf{X}, A_l\right]. \tag{7.5}$$

For $k = 1$,

$$\begin{aligned} E_{\mathbf{X}|A_l}\left[r_l\right] &= \int r_l f(\mathbf{X}|A_l) d\mathbf{X} = \int \sum_{j=1}^{J} R_{Label(l),j} P\left[C_j|\mathbf{X}, A_l\right] f(\mathbf{X}|A_l) d\mathbf{X} \\ &= \sum_{j=1}^{J} R_{Label(l),j} P\left[C_j|A_l\right], \qquad (7.6) \\ E\left[r_l\right] &= E_{\mathbf{P}}\left[E_{\mathbf{X}|A_l}\left[r_l\right]\right] = \sum_{j=1}^{J} R_{Label(l),j} E_{\mathbf{P}}\left[P\left[C_j|A_l\right]\right]. \qquad (7.7) \end{aligned}$$

According to (3.19),

$$E_{\mathbf{P}}\left[P\left[C_j|A_l\right]\right] = \frac{M_{j,l}}{M_l}, \tag{7.8}$$

$$E\left[r_l\right] = \sum_{j=1}^{J} R_{Label(l),j} \frac{M_{j,l}}{M_l}. \tag{7.9}$$

where $M_{j,l} = n_{j,l} + \lambda$, $M_l = n_l + \lambda J = \sum_{j=1}^{J} M_{j,l}$.

To compute the k-th moment of r for $k > 1$, we define

$$r_l^k = \left(\sum_{j=1}^{J} R_{Label(l),j} P\left[C_j|\mathbf{X}, A_l\right] \right)^k, \tag{7.10}$$

instead of

$$r_l^k = \sum_{j=1}^{J} R^k_{Label(l),j} P\left[C_j|\mathbf{X}, A_l\right], \tag{7.11}$$

because in case of uniform costs, $R_{i,j}$ is either 0 or 1, and $R^k_{i,j} \equiv R_{i,j}$, which means that the pruning algorithm using (7.11) always produces the same pruned tree as $k = 1$ that tends to be the same as the fully grown tree.

According to (3.21), for $k = 2$,

$$\begin{aligned}
E\left[r_l^2\right] &= \sum_{i,j} R_{Label(l),i} R_{Label(l),j} E\left[P[C_i|\mathbf{X}, A_l] P[C_j|\mathbf{X}, A_l]\right] \\
&\approx \sum_{i,j} R_{Label(l),i} R_{Label(l),j} E_{\mathbf{P}}\left[E_{\mathbf{X}}\left[P[C_i|\mathbf{X}, A_l]\right] E_{\mathbf{X}}\left[P[C_j|\mathbf{X}, A_l]\right]\right] \\
&= \sum_{i,j} R_{Label(l),i} R_{Label(l),j} E_{\mathbf{P}}\left[P[C_i|A_l] P[C_j|A_l]\right] \\
&= \sum_{i,j} R_{Label(l),i} R_{j,Labe(l)} \frac{M_{i,l}\left(M_{j,l} + I\left[i = j\right]\right)}{M_l(M_l + 1)} \\
&= \frac{\sum_{i,j} R_{Label(l),i} R_{Label(l),j} M_{i,l} M_{j,l} + \sum_j R^2_{Label(l),j} M_{j,l}}{M_l(M_l + 1)} \\
&= \frac{\left(\sum_j R_{Label(l),j} M_{j,l}\right)^2 + \sum_j R^2_{Label(l),j} M_{j,l}}{M_l(M_l + 1)}
\end{aligned} \tag{7.12}$$

Note that although $P[C_i|\mathbf{X}, A_l]$ and $P[C_j|\mathbf{X}, A_l]$ are actually not independent, the above approximation is necessary because we cannot estimate their product for all possible $\mathbf{X}$.

For a general value of k,

$$\begin{aligned} E\left[r_l^k\right] &= E\left[\left(\sum_{j=1}^{J} R_{Label(l),j} P\left[C_j|\mathbf{X}, A_l\right]\right)^k\right] \\ &\approx E_{\mathbf{P}}\left[\left(\sum_{j=1}^{J} R_{Label(l),j} P\left[C_j|A_l\right]\right)^k\right]. \end{aligned} \tag{7.13}$$

Let E_k^* denote the right hand side of the above equation. E_k^* can be evaluated by recursion:

$$E_k^* = \sum_{m=1}^{k} \frac{\frac{(k-1)!}{(k-m)!} E_{k-m}^* \sum_{i=1}^{J} M_{i,l} R_{Label(l),i}^m}{\prod_{i=1}^{m}(M_l + k - m)} \tag{7.14}$$

$$E_0^* = 1. \tag{7.15}$$

Please refer to Appendix C.9 for the proof. So far we are unable to simplify (7.14), whose computational complexity is quadratic to k.

It worth mentioning that to minimize the misclassification cost $Label(l)$ should not be simply chosen as the majority class as in (3.8). Instead, we should set

$$Label(l) = \arg\min\ E\left[r_l^k\right]. \tag{7.16}$$

Unlike the case of minimizing the error rate, now the label depends on both k and λ.

7.2 Error of Class Probabilities

It has been observed that decision trees can also estimate the class probabilities by simply outputting the class proportions of the leaf chosen to represent the input. The decision trees that perform this estimation are called Probability Estimator Trees (PETs). This estimate of PET, turns out to be, in most cases, very poor (see [Bra97]). Being aware of the drawback of the maximum likelihood estimate, Cestnik suggested using the Laplace's (or Lidstone's) estimation in Machine Learning for probability estimation (see [Ces90]). The utilization of Lidstone's estimation turned out successful in class probabilities prediction and related problems such as misclassification cost minimization (see [FFH03, PD03, PMM94]). Ferri et al. also applied the m-Estimate and their m-Branch to smooth the probabilities (see [FFH03]). These results are not surprising, because Lidstone's estimation represents the optimal posterior probability estimate assuming certain priors.

In this section, we apply our k-norm estimation to evaluate how well a tree predicts the class probabilities. Same as before, a by-product is a pruning algorithm that prunes a PET efficiently.

7.2.1 Estimation for sub-trees

Our goal is to estimate the error of the class probabilities, defined below:

$$r_{j,t} = P\left[C_j|\mathbf{X},A_t\right] - P^*\left[C_j|\mathbf{X},A_t\right] \tag{7.17}$$

Applying Theorem 7.1 with $H = 2, a_1 = 1, a_2 = -1, P\left[\Phi_1|A_d,\mathbf{X}\right] = P\left[C_j|\mathbf{X},A_d\right]$, and $P\left[\Phi_2|A_d,\mathbf{X}\right] = P^*\left[C_j|\mathbf{X},A_d\right]$, we have

$$E\left[r_{j,d}^k\right] = \sum_{c\in Children(d)} E\left[r_{j,c}^k\right] E_{\mathbf{P}}\left[P\left[A_c|A_d\right]\right]. \tag{7.18}$$

Note that odd values of k may result in a negative number, and they are not very useful. For example, since $E[r_{j,l}] = 0$, $\sigma[r_{j,l}] = \sqrt{E[r_{j,l}^2]}$ and the output "0 ± 0.1" is not quite meaningful. Instead, we can output the mean-square-error $E[r_{j,l}^2]$ together with the standard deviation of the square-error $\sigma[r_{j,l}^2] = \sqrt{E[r_{j,l}^4] - E[r_{j,l}^2]^2}$.

To consider all classes in the pruning algorithm, we can combine the moments of the errors among all classes as $r_t = \sqrt[k]{\sum_{j=1}^{J} E\left[r_{j,t}^k\right]}$ rather than $r_t = \sum_{j=1}^{J} \sqrt[k]{E\left[r_{j,t}^k\right]}$, so that the error of the branches can still sum up independently as in (4.10), which means that the pruning algorithm can still finish in one traversal of the tree.

7.2.2 Estimation for leaves

For a leaf l,

$$\begin{aligned} E\left[r_{j,l}^{k}\right] &\approx E_{\mathbf{P}}\left[\left(P\left[C_j|A_l\right]-P^*\left[C_j|A_l\right]\right)^k\right] \\ &= \sum_{m=0}^{k}\frac{(k)!(-1)^m}{m!\,(k-m)!}E\left[P[C_j|A_l]^m\right]P^*[C_j|A_l]^{k-m} \end{aligned} \quad (7.19)$$

$E\left[P[C_j|A_l]^m\right]$ is evaluated with (3.19). Thus,

$$E\left[r_{j,l}^{k}\right]\approx\sum_{m=0}^{k}\frac{(k)!(-1)^m}{m!\,(k-m)!}\prod_{i=0}^{m-1}\frac{n_{j,l}+\lambda+i}{n_l+\lambda J+i}\left(\frac{n_{j,l}+\lambda}{n_l+\lambda J}\right)^{k-m} \quad (7.20)$$

For example ($k=2,4$),

$$E\left[r_{j,l}^{2}\right]\approx\frac{M_{j,l}(M_l-M_{j,l})}{M_l^2(M_l+1)} \quad (7.21)$$

$$E\left[r_{j,l}^{4}\right]\approx\frac{3M_{j,l}(M_l-M_{j,l})(M_l^2M_{j,l}-M_lM_{j,l}^2+6M_{j,l}^2-6M_{j,l}M_l+2M_l^2)}{M_l^4(M_l+1)(M_l+2)(M_l+3)} \quad (7.22)$$

where $M_{j,l}=n_{j,l}+\lambda$, $M_l=n_l+\lambda J=\sum_{j=1}^{J}M_{j,l}$.

Unfortunately we cannot provide a simple expression for $E\left[r_{j,l}^{k}\right]$ in a closed form (without summation) for any value of k. Nevertheless, we know that given a set of instances of a random variable, the estimated higher order moments are less reliable. For this reason, we recommend $k=2$ (representing the mean-square-error).

CHAPTER 8
SUMMARY AND FUTURE RESEARCH

8.1 Summary

In this dissertation, we focused on the error rate estimation in decision tree classifiers. We showed the lack of theoretical justification of the error rate estimation employed in current tree pruning algorithms. We applied Lidstone's law of succession to our estimation, but differing from the existing approaches that simply use the expected values, our analysis is based on the underlying Dirichlet distribution so that we are able to provide a closed-form expression for all the moments of the error rates of tree leaves. We have completed the derivation of overall tree error rates by proving that under a loose constraint, any moment of a sub-tree's error rate can be computed as the summation of those of its branches. By computing the moments of the error rate, we are able to provide other statistical characteristics of the estimate, such as the standard deviation of the estimate quantifying the reliability of the estimate (expected value) of the error rate.

Based on the tree error rate estimation we proposed a k-norm pruning algorithm as a generalized version of the existing MEP algorithm (which represents the special case of $k = 1$). We proved that MEP tends to under-prune a tree, while our k-norm pruning algorithm for $k \geq 2$ does not suffer from the same shortcoming, since it takes into account both the expected value and the standard deviation of the error.

We provided a thorough analysis on the properties of our k-norm pruning algorithm. We showed that our algorithm has the desirable properties, as follows. First, when the parameter is k fixed it can finish the pruning within one traversal (with time complexity $\Theta(L)$) of the fully grown tree without examining all $\Theta(1.5^L)$ possible pruned trees (L is the number of leaves in the fully grown tree). Second, when the parameter k varies the resulting pruned trees can be sorted in a pruning sequence, which can be found and evaluated efficiently, with typical total time complexity $\Theta(L \log L) + \Theta(N \log L)$ where N is the size of the validation data set. Third, the error rate estimate can be controlled by the parameter k (higher k produces more pessimistic estimates). Fourth, the k-norm pruning algorithm removes any ineffective split for any natural number k. All of the above properties emulate the CCP properties of CART, which is a well established and theoretically sound pruning procedure. Furthermore, we showed that our algorithm possesses some unique (compared to CCP of CART) important characteristics: (a) its parameter k can be preset to 2 in order to avoid validation and to further reduce the time complexity to $\Theta(L)$, (b) its k-norm estimation has a clear theoretical interpretation and the case $k = \infty$ can be well-explained, and (c) it provides protection against pruning of useful splits.

We performed a thorough and empirical comparison of our k-norm pruning algorithm (for $k = 2$) against two other classical pruning algorithms (CCP and EBP). The results showed that 2-norm is better in accuracy than CCP 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 any additional data (training or validation) during the pruning phase. In particular, our experimental results demonstrated that our 2-norm algorithm is orders of magnitude faster than CCP and EBP. Specifically, when we experimented with 2-norm, CCP and EBP algorithms for very large databases (half a million patterns) the time required to generate the pruned tree was tenths of a second for the 2-norm, versus hundreds of seconds for EBP and thousands of seconds for CCP. These experiments with large databases justified our claim that 2-norm scales better, as the database size increases, than EBP or CCP. Finally, in comparing the 2-norm pruning algorithm with other pruning strategies (in addition to CCP and EBP) we concluded that 2-norm compares favorably with a multitude of other pruning strategies that have appeared in the literature.

This dissertation contributes to the literature in the following aspects: 1) We enhanced the estimation methodology by providing explicit expressions for estimating the error rate and for evaluating the reliability of the estimate. Since the actual error rate is unknown and the estimate cannot be 100% accurate, the reliability is apparently helpful when performing automatic decisions. 2) We proposed a k-pruning algorithm based on our estimation of the statistical characteristics of the error rate (average value and standard deviation). This algorithm has a clear theoretical interpretation and a series of desirable, good properties. It is accurate even when the size of the training data set is small, and it is very fast even when the size of the training data set is huge.

8.2 Future Research

It is important to note that k-norm estimations can be generalized, as follows.

8.2.1 Other definitions of tree risks

Theorem 4.1 can be generalized to other measures of interests. For example, the risks can be redefined using appropriate misclassification costs or by using the MSE of class probabilities, as shown in Chapter 7. The prediction of these risks is not extensively studied, and the only well-known approaches are cross-validation and the use of the expected values. We have listed all the necessary equations for the k-norm risk estimation in these domains.

Similar to the error rates, we are able to compute the standard deviation to indicate the reliability of the estimate, and we can adapt our k-norm pruning algorithm to output the pruned tree that minimizes the k-norm risk rather than the k-norm error rate. The adapted k-norm pruning algorithm will inherit some of the properties that we reported for the error rate, such as that it can finish pruning within only one tree traversal. Whether or not the other properties are applicable, as well as of how to choose the parameters for the adapted pruning algorithm (e.g., λ), remains an open problem that requires further research.

8.2.2 Extensions of conditions

In our derivations, equations (4.28) and (7.18) for estimating the risks (error rate, misclassification cost and error of class probability) of a sub-tree is based on the *Partitioning Assumption* and the *Determinacy Assumption*; equations (4.29), (7.13) and (7.20) for estimating the risks of a leaf are based on the *Piece-wise Independence Assumption.* It would be interesting to see how to derive the equations for the cases where one or more of the assumptions are violated. Following are some examples of these cases.

8.2.2.1 tree ensembles

Ensemble learning is one of the most attractive trends in current Machine Learning research. It has been shown in [Die00, Bre01] that an ensemble tree can outperform an individual tree significantly in accuracy, at the cost of size, speed, and interpretability. To estimate the risk of a tree ensemble, we can view the ensemble as a tree with a higher level and each individual tree classifier as a branch. In this case, all branches are activated when an input is given, thus violating the *Partitioning Assumption.* If the estimated class probabilities (or scores) can be treated as a linear combination among those of the branches and each branch has a constant weight, we can normalize the weights and treat them as activation probabilities (by considering each branch as being activated at a probability), and thus the violation of the *Partitioning Assumption* is transformed to that of the *Determinacy Assumption* (see below).

8.2.2.2 missing values

If a decision node d has a split test "$x_1 \leq 0$" (that is, $P[A_{c_1}|A_d, \mathbf{X}] = I[x_1 \leq 0]$), but x_1 is missing from the input $\mathbf{X}$, then $P[A_{c_1}|A_d, \mathbf{X}]$ is unknown. The algorithms that handle missing values usually approximate $P[A_{c_1}|A_d, \mathbf{X}]$ by another known quantity. For example, C4.5 (see [Qui93]) uses $P[A_{c_1}|A_d]$ as an approximation. In this case, each child of d is activated with a certain probability, introducing two issues for our k-norm risk estimation:

First, the above handling of missing values leads into a violation of the *Determinacy Assumption*. For this case, it may be difficult to derive a closed-form expression to estimate the k-norm risks for all k, but it is still possible for $k = 2$ and for binary trees.

Secondly, in Lidstone's law of succession and Dirichlet distribution, the number of occurrences n_h is assumed to be a deterministic integer, as implied in the computation of $P[\{n_h\}|\{p_h\}]$ in (3.17). If an example is covered by a leaf l at a certain probability due to missing data, how to count $n_{j,l}$ has to be addressed before applying (4.29). Strictly speaking, $n_{j,l}$ becomes a random number and it is no longer our observation $\mathbf{O}$; the theorem of total probability $f(\{p_h\}|\mathbf{O}) = \sum_{n_1+\ldots+n_H=N} f(\{p_h\}|\{n_h\})P[\{n_h\}|\mathbf{O}]$ should be applied, together with (3.16). The expression may be very difficult to simplify due to the many terms on the right hand side, which means we may not be able to obtain closed-form expressions for the posterior k-norm risk estimates. An alternative is to replace $n_{j,l}$ with its expected value $E[n_{j,l}] = \sum_i P[A_l, C_j|\mathbf{X}_i]$ (where $\mathbf{X}_i$ is the i-th training example) in (4.29), (7.13) and (7.20). This approximation, however, requires more justification because (3.17) is by-passed.

8.2.2.3 flexible in-leaf predictions

Smyth et al., questioned the validity of outputting the same class probabilities for all inputs that activate the same leaf (see [SGF95]). They showed the advantage of their modification on the tree prediction algorithm by incorporating kernel functions to estimate the class probabilities, which become dependent on the input attributes even when the leaf is determined. In this case, the *Piece-wise Independence Assumption* is violated. Since Lidstone's law of succession relies only on the number of examples in the region covered by the leaf, not on the distribution of these examples, the risk estimate of leaves must be calculated differently.

8.2.3 Other classifiers

Furthermore, Theorem 4.1 can be applied to any other classifiers beyond decision trees that iteratively create nodes to represent subregions of the input space, such as Fuzzy ARTMAP (see [CGM92]). It is interesting to see if a similar pruning algorithm can be applied to these classifiers, e.g., to address the category proliferation problem of Fuzzy ARTMAP, a problem that has been studied by many researchers in the field (see [MH95, CKG01, DK01] for example). If so, the properties and the parameter settings are, again, potential topics for future research.

APPENDIX A
NOTATIONS

Table A.1: Frequently Used Notations Listed in Alphabetical Order

Symbol	Definition	Constraints or properties
A_t	The event that the node t is activated (receives an input)	A_t implies $A_{Parent(t)}$
$b_t = n_{j,t} - \max\limits_j n_{j,t}$	Number of misclassified training examples in node t	$0 \le \frac{b_t}{n_t} \le \frac{B_t}{M_t} \le \frac{J-1}{J}$
$B_t = b_t + (J-1)\lambda$	Numerator in Lidstone's Estimate of the error rate in node t	$0 \le \frac{b_t}{n_t} \le \frac{B_t}{M_t} \le \frac{J-1}{J}$
$Children(d)$	The set of immediate children nodes of a decision node d	$K_d = \|Children(d)\| \ge 2$
$E[Q]$	Expected value of a random variable Q	
η	The smoothing parameter for the estimation of the node proportions	$\eta \ge 0$
$f(\mathbf{X})$	Joint probability density function of the attributes in $\mathbf{X}$	$f(\mathbf{X}) \ge 0$, $\int f(\mathbf{X})d\mathbf{X} = 1$
$\Gamma(x)$	Gamma function	$\frac{\Gamma(x+k)}{\Gamma(x)} = \Pi_{i=0}^{k-1}(x+i)$
J	Number of classes	$J \ge 2$
K_d	Number of immediate children of a decision node d	$K_d = \|Children(d)\| \ge 2$
$Label(t)$	Predicted class label of leaf t	$Label(t) = \arg\max\limits_j n_{j,t}$
$Leaves(t)$	The set of leaf nodes in the sub-tree rooted at node t	
λ	The smoothing parameter for class probability estimation	$\lambda \ge 0$
$M_{j,t} = n_{j,t} + \lambda$	Numerator in Lidstone's Estimate of the class probabilities in node t	$M_{j,t} \ge \lambda$, $\sum_{j=1}^J M_{j,t} = M_t$
$M_t = n_t + J\lambda$	Denominator in Lidstone's Estimate of the error/class probabilities in node t	$J\lambda < M_t < M_{Parent(t)}$
$n_{j,t}$	Number of examples of class j passed to node t	$n_{j,t} \ge 0$, $\sum_{j=1}^J n_{j,t} = n_t$
$n_t = \sum_{j=1}^J n_{j,t}$	Number of examples passed to node t	$n_t > 0, n_t \ge n_{j,t}$
$\mathbf{O}$	The observations with training examples and the tree classifier	
$P[\Phi]$	Probability of event Φ	$0 \le P[\Phi] \le 1$
$Parent(t)$	The parent node (immediate ancestor) of node t	
$p_{c\|d} = E[P[A_c\|A_d]\|\mathbf{O}]$	Proportion of node c in the sub-tree rooted at d	$\sum_{c\in Leaves(d)} p_{c\|d} = 1$
$\Psi(x)$	Digamma function	$\Psi(x) = \frac{d}{dx}\log\Gamma(x)$
$\\|Q\\|_k = \sqrt[k]{E\left[\|Q\|^k\|\mathbf{O}\right]}$	k-norm of the random variable Q	
$R_{i,j}$	Risk of classifying $\mathbf{X}$ as belonging to class i while the true class label is j	$R_{i,j} \ge 0$
$\sigma[Q] = \sqrt{E\left[Q^2\right] - E[Q]^2}$	Average standard deviation of random variable Q	$\sigma \ge 0$
$\mathbf{X}$	A training example or an unseen input, represented by a vector of attributes	

APPENDIX B
A CASE STUDY ON IRIS DATABASE

The *Iris* database is extensively examined in the literature. It can be downloaded from the UCI repository (see [NHB98]). 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 B.1(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.

B.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 Figure B.1(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 4.3 for the definition of effective splits).

B.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 compare the pruning sequence of CCP and that generated by our k-norm pruning algorithm with various k.

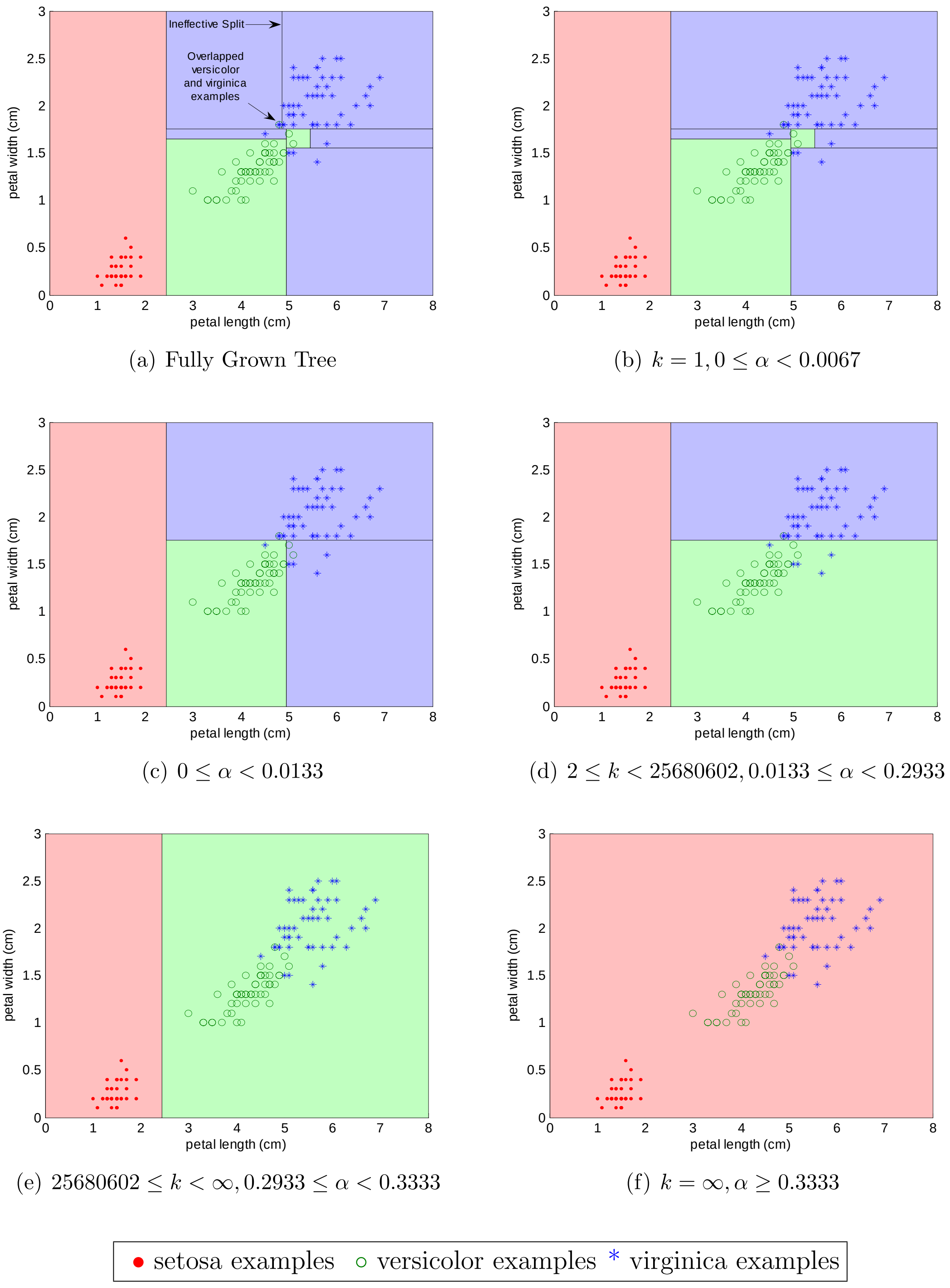
By increasing α from 0, CCP yielded a pruning sequence of 5 trees, each tree is a further pruned version of the previous tree (the 0-th tree is the fully grown tree). By raising k from 1, the k-norm pruning algorithm yielded a pruning sequence of 4 trees (proven by Theorem 5.3). These 4 trees also appeared in CCP's pruning sequence, as shown in Figure B.1.

The tree pruned with $k = 2$ (see Figure B.1(d) and Figure B.2) can be evaluated as follows. For the leaf c_{21},

$$\begin{aligned} E\left[r_{c_{21}}\right] &= \frac{5+0.5\times 2}{54+0.5\times 3} = 0.1081, & \text{(B.1)}\\ E\left[r_{c_{21}}^2\right] &\approx \frac{6\times 7}{55.5\times 56.5} = 0.01339, & \text{(B.2)}\\ P^*\left[A_{c_{21}}|A_{c_2}\right] &= \frac{54+0.5}{100+0.5\times 2} = 0.5396. & \text{(B.3)} \end{aligned}$$

For the leaf c_{22},

$$\begin{aligned} E\left[r_{c_{22}}\right] &= \frac{1+0.5\times 2}{46+0.5\times 3} = 0.04211, & \text{(B.4)}\\ E\left[r_{c_{22}}^2\right] &\approx \frac{2\times 3}{47.5\times 48.5} = 0.002604, & \text{(B.5)}\\ P^*\left[A_{c_{22}}|A_{c_2}\right] &= \frac{46+0.5}{100+0.5\times 2} = 0.4604. & \text{(B.6)} \end{aligned}$$



(a) Fully Grown Tree

(b) $k = 1, 0 \leq \alpha < 0.0067$

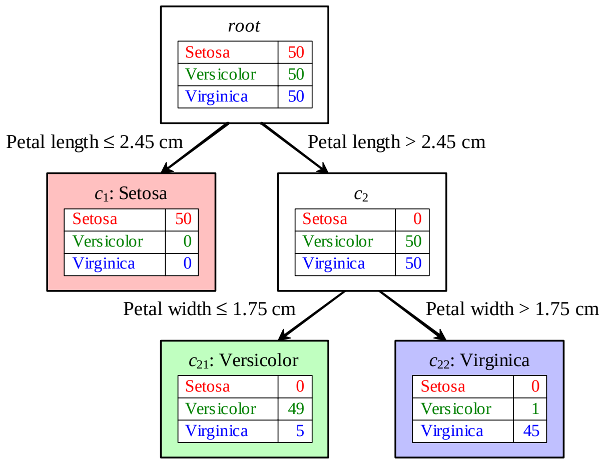
(c) $0 \leq \alpha < 0.0133$

(d) $2 \leq k < 25680602, 0.0133 \leq \alpha < 0.2933$

(e) $25680602 \leq k < \infty, 0.2933 \leq \alpha < 0.3333$

(f) $k = \infty, \alpha \geq 0.3333$

Figure B.1: Splits in Pruning Sequence for *Iris* Database



Note: 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). The decision regions are plotted in Figure B.1(d).

Figure B.2: Tree Grown with *Iris* Database and Pruned with 2-Norm Criterion

For the decision node c_2,

$$E\left[r_{c_2}\right] = E\left[r_{c_{21}}\right] P^*\left[A_{c_{21}} | A_{c_2}\right] + E\left[r_{c_{22}}\right] P^*\left[A_{c_{22}} | A_{c_2}\right] = 0.07772, \tag{B.7}$$

$$E\left[r_{c_2}^2\right] = E\left[r_{c_{21}}^2\right] P^*\left[A_{c_{21}} | A_{c_2}\right] + E\left[r_{c_{22}}^2\right] P^*\left[A_{c_{21}} | A_{c_2}\right] \approx 0.008427, \tag{B.8}$$

$$P^*\left[A_{c_2} | A_{root}\right] = \frac{100 + 0.5}{150 + 0.5 \times 2} = 0.6656. \tag{B.9}$$

For the leaf c_1,

$$E\left[r_{c_1}\right] = \frac{0 + 0.5 \times 2}{50 + 0.5 \times 3} = 0.01942, \tag{B.10}$$

$$E\left[r_{c_1}^2\right] \approx \frac{1 \times 2}{51.5 \times 52.5} = 0.0007397, \tag{B.11}$$

$$P^*\left[A_{c_1} | A_{root}\right] = \frac{50 + 0.5}{150 + 0.5 \times 2} = 0.3344. \tag{B.12}$$

For the root (representing the entire tree)

$$E\left[r_{root}\right] = E\left[r_{c_1}\right] P^*\left[A_{c_1}|A_{root}\right] + E\left[r_{c_2}\right] P^*\left[A_{c_2}|A_{root}\right] = 0.05822, \tag{B.13}$$

$$E\left[r_{root}^2\right] = E\left[r_{c_1}^2\right] P^*\left[A_{c_1}|A_{root}\right] + E\left[r_{c_2}^2\right] P^*\left[A_{c_2}|A_{root}\right] \approx 0.005856, \tag{B.14}$$

$$\sigma\left[r_{root}\right] = \sqrt{E\left[r_{root}^2\right] - E\left[r_{root}\right]^2} \approx 0.04966, \tag{B.15}$$

$$\left\|r_{root}\right\|_2 = \sqrt{E\left[r_{root}^2\right]} = 0.07652, \tag{B.16}$$

where $E\left[r_{root}\right]$ is the expected value of the error rate and the standard deviation 0.04966 measures the uncertainty of the error rate; $\left\|r_{root}\right\|_2$ 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.

B.3 Tree Prediction

We use the tree pruned with $k = 2$ (see Figure B.1(d) and Figure B.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:

$$E_{\mathbf{P}}\left[r\right] \approx E\left[r_{c_1}\right] = 0.01942, \tag{B.17}$$

$$\sigma_{\mathbf{P}}\left[r\right] \approx \sqrt{E\left[r_{c_1}^2\right] - E\left[r_{c_1}\right]^2} = 0.0190. \tag{B.18}$$

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

$$\|r\|_2 \approx \sqrt{E\left[r_{c_1}^2\right]} = 0.0272. \tag{B.19}$$

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:

$$\begin{aligned} E_{\mathbf{P}}[r] &\approx E\left[r_{c_{21}}\right] = 0.1081, && \text{(B.20)} \\ \sigma_{\mathbf{P}}[r] &\approx \sqrt{E\left[r_{c_{21}}^2\right] - E\left[r_{c_{21}}\right]^2} = 0.04131, && \text{(B.21)} \\ \|r\|_2 &\approx \sqrt{E\left[r_{c_{21}}^2\right]} = 0.1157. && \text{(B.22)} \end{aligned}$$

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:

$$\begin{aligned} E_{\mathbf{P}}[r] &\approx E\left[r_{c_{22}}\right] = 0.04211, && \text{(B.23)} \\ \sigma_{\mathbf{P}}[r] &\approx \sqrt{E\left[r_{c_{22}}^2\right] - E\left[r_{c_{22}}\right]^2} = 0.02884, && \text{(B.24)} \\ \|r\|_2 &\approx \sqrt{E\left[r_{c_{22}}^2\right]} = 0.05103. && \text{(B.25)} \end{aligned}$$

APPENDIX C
PROOFS OF THEOREMS AND EQUATIONS

C.1 Proof of Equations (3.19), (3.20) and (3.21)

we first list some important properties of Dirichlet distribution. If

$$(p_1, \dots, p_H) \sim Dir(\lambda_1, \dots, \lambda_H), \tag{C.1}$$

then the following are true.

1. The posterior probabilities also follow the Dirichlet distribution (see [Goo65]). In particular,

$$(p_1, \dots, p_H | n_1, \dots, n_H) \sim Dir(n_1 + \lambda_1, \dots, n_H + \lambda_H). \tag{C.2}$$

2. It has also been shown (see [GCS95], page 482) that

$$(p_h, 1 - p_h) \sim Dir\left(\lambda_h, \sum_{m \neq h} \lambda_m\right). \tag{C.3}$$

3. It can be easily proven that the product of the probabilities has the following expected value (see [BN03], page 274):

$$E\left[p_1^{k_1} \dots p_H^{k_H}\right] = \frac{\frac{\Gamma\left(\sum_{h=1}^H \lambda_h\right)}{\Gamma\left(\sum_{h=1}^H (\lambda_h + k_h)\right)}}{\prod_{h=1}^H \frac{\Gamma(\lambda_h)}{\Gamma(\lambda_h + k_h)}}, \tag{C.4}$$

where $k_1, \ldots, k_H$ are non-negative integers. Using the following property of $\Gamma()$ function:

$$\frac{\Gamma(x+k)}{\Gamma(x)} = \prod_{m=0}^{k-1} (x+m), \tag{C.5}$$

we have

$$E\left[p_1^{k_1} \ldots p_H^{k_H}\right] = \frac{\prod_{h=1}^{H} \prod_{m=0}^{k_h-1} (\lambda_h + m)}{\prod_{m=0}^{k-1} (\sum_{h=1}^{H} \lambda_h + m)}, \tag{C.6}$$

where $k = \sum_{h=1}^{H} k_h$. As two special cases,

$$E\left[p_h^k\right] = \prod_{m=0}^{k-1} \frac{\lambda_h + m}{\sum_{h=1}^{H} \lambda_h + m}, \tag{C.7}$$

$$E\left[p_{h_1} p_{h_2}\right] = \frac{\lambda_{h_1} (\lambda_{h_2} + I[h_1 = h_2])}{\sum_{h=1}^{H} \lambda_h \left(1 + \sum_{h=1}^{H} \lambda_h\right)}. \tag{C.8}$$

Using (C.2) and (C.7), we prove (3.19); using (C.2), (C.3) and (C.7), we prove (3.20); using (C.2) and (C.8), we prove (3.21). $\blacksquare$

C.2 Proof of Theorem 4.1

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

$$E_{\mathbf{X}|A_d}\left[r_d^k\right] = \sum_{c \in Children(d)} E_{\mathbf{X}|A_c}\left[r_c^k\right] P\left[A_c | A_d\right], \tag{C.9}$$

where

$$r_d = P\left[\Phi|A_d, \mathbf{X}\right], \tag{C.10}$$

given that the events $\{A_c|A_d\}_{c\in Children(d)}$ are mutually exclusive and collectively exhaustive and that $P\left[A_c|A_d, \mathbf{X}\right]$ 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,

$$P\left[\Phi|A_d, \mathbf{X}\right] = \sum_{c\in Children(d)} P\left[\Phi|A_c, \mathbf{X}\right] P\left[A_c|A_d, \mathbf{X}\right], \tag{C.11}$$

that is,

$$r_d = \sum_{c\in Children(d)} r_c P\left[A_c|A_d, \mathbf{X}\right]. \tag{C.12}$$

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

$$P\left[A_{c_1}|A_d, \mathbf{X}\right] P\left[A_{c_2}|A_d, \mathbf{X}\right] = 0,\ if\ c_1 \neq c_2, \tag{C.13}$$

$$P\left[A_c|A_d, \mathbf{X}\right]^k = P\left[A_c|A_d, \mathbf{X}\right], \tag{C.14}$$

and therefore,

$$r_d^k = \sum_{c\in Children(d)} r_c^k P\left[A_c|A_d, \mathbf{X}\right], \tag{C.15}$$

$$\begin{aligned} E_{\mathbf{X}|A_d}\left[r_d^k\right] &= \int r_d^k f(\mathbf{X}|A_d) d\mathbf{X} \\ &= \sum_{c\in Children(d)} \int \left[r_c^k\right] P\left[A_c|A_d, \mathbf{X}\right] f(\mathbf{X}|A_d) d\mathbf{X}. \end{aligned} \tag{C.16}$$

According to Bayesian Theory,

$$P\left[A_c|A_d,\mathbf{X}\right]f(\mathbf{X}|A_d)=f(\mathbf{X}|A_c,A_d)P\left[A_c|A_d\right]=f(\mathbf{X}|A_c)P\left[A_c|A_d\right], \tag{C.17}$$

and therefore,

$$\begin{aligned} E_{\mathbf{X}|A_d}\left[r_d^k\right] &= \sum_{c\in Children(d)}\int\left[r_c^k\right]f(\mathbf{X}|A_c)d\mathbf{X}P\left[A_c|A_d\right] \\ &= \sum_{c\in Children(d)}E_{\mathbf{X}|A_c}\left[r_c^k\right]P\left[A_c|A_d\right]. \end{aligned} \tag{C.18}$$

∎

C.3 Proof of Theorem 4.3

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

$$r_{leaf}(t)=\frac{b_t+(J-1)\lambda}{n_t+J\lambda}, \tag{C.19}$$

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,

$$r_{tree}(t)=\sum_{c\in Children(t)}r_{leaf}(c)\frac{n_c+\eta}{n_t+K_t\eta}. \tag{C.20}$$

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.

$$\begin{aligned} \frac{\partial}{\partial \eta} \sum_c \frac{(b_c + (J-1)\lambda)(n_c + \eta)}{(n_c + J\lambda)(n_t + K_t \eta)} &= \sum_c \frac{(b_c + (J-1)\lambda)(n_t - K_t n_c)}{(n_c + J\lambda)(n_t + K_t \eta)^2} \\ &= \frac{q}{(n_t + K_t \eta)^2}. \end{aligned} \tag{C.21}$$

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 $\eta = 0$:

$$\begin{aligned}
r_{tree}(t) &= \frac{1}{n_t} \sum_{c \in Children(t)} \frac{n_c\,(b_c + (J-1)\lambda)}{n_c + J\lambda} \\
&< \frac{1}{n_t} \sum_{c \in Children(t)} \frac{n_t\,(b_c + (J-1)\lambda)}{n_t + J\lambda} \\
&\leq \frac{b_t + K_t(J-1)\lambda - 1}{n_t + J\lambda} \\
&\leq \frac{b_t + K_t(J-1)\lambda - (K_t - 1)(J-1)\lambda}{n_t + J\lambda} \\
&= \frac{b_t + (J-1)\lambda}{n_t + J\lambda} \\
&= r_{leaf}(t). \qquad (C.22)
\end{aligned}$$

Case $\eta = J\lambda$:

$$\begin{aligned}
r_{tree}(t) &= \frac{\sum_{c \in Children(t)} (b_c + (J-1)\,\lambda)}{n_t + K_t J\lambda} \\
&\leq \frac{b_t - 1 + (J-1)\,K_t\lambda}{n_t + K_t J\lambda} \\
&< \frac{b_t}{n_t + J\lambda} + \frac{(J-1)\,K_t\lambda - 1}{n_t + K_t J\lambda}. \qquad (C.23)
\end{aligned}$$

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

$$\frac{(J-1)\,K_t\lambda - 1}{n_t + K_t J\lambda} < \frac{(J-1)\lambda}{n_t + J\lambda}, \qquad (C.24)$$

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

C.4 Proof of Theorem 5.3

The theorem states that if a sub-tree does not decrease the k-norm error rate compared to a leaf, it increases the $(k+1)$-norm error rate. To express the theorem mathematically, we define two functions:

$$g_k(x) = \frac{\Gamma(x+k)}{\Gamma(x)} = \prod_{i=0}^{k-1}(x+i), \tag{C.25}$$

$$f_k(B, M) = \frac{g_k(B)}{g_k(M)} = \frac{\Gamma(B+k)\Gamma(M)}{\Gamma(B)\Gamma(M+k)}. \tag{C.26}$$

For a leaf l, $f_k(B_l, M_l)$ represents the k-th moment of the error rate in l. Please refer to Appendix A for the definitions of the symbols. The theorem can now be expressed as follows:

Assume $k \geq 1$ and $0 < B_c < M_c < M_d$ for any leaf c under a decision node d. If

$$\sum_{c \in Leaves(d)} f_k(B_c, M_c) p_{c|d} \geq f_k(B_d, M_d), \tag{C.27}$$

then

$$\sum_{c \in Leaves(d)} f_{k+1}(B_c, M_c) p_{c|d} > f_{k+1}(B_d, M_d). \tag{C.28}$$

Proof To prove this theorem, we find two values between the left hand side and the right hand side of (C.28). We prove that they are in between by two lemmas.

C.4.1 Proof of Lemma 5.1

This lemma is proposed in subsection 5.1.2. It states that if two leaves have the same k-norm error rate, the smaller leaf have higher $(k+1)$-norm error rate. That is, If there exists a value s, such that

$$f_k(B_c, M_c) = f_k(s, M_d), \tag{C.29}$$

and $0 < B_c < M_c < M_d$, then $f_{k+1}(B_c, M_c) > f_{k+1}(s, M_d)$.

Assume s and M_d are fixed. B_c is a function of M_c due to (C.29), and so is $f_{k+1}(B_c, M_c)$. Note that once M_c is given, there is a unique solution for B_c in $(0, M_c)$. Now we prove that $f_{k+1}(B_c, M_c)$ is a strictly decreasing function of M_c. According to (C.29),

$$\frac{\Gamma(B_c + k)}{\Gamma(B_c)} = f_k(s, M_d)\frac{\Gamma(M_c + k)}{\Gamma(M_c)}. \tag{C.30}$$

Taking the derivative of both sides with respect to M_c,

$$\frac{d}{dB_c}\left(\frac{\Gamma(B_c + k)}{\Gamma(B_c)}\right)\frac{dB_c}{dM_c} = f_k(s, M_d)\frac{d}{dM_c}\left(\frac{\Gamma(M_c + k)}{\Gamma(M_c)}\right), \tag{C.31}$$

$$\begin{aligned}\frac{dB_c}{dM_c} &= \frac{f_k(s, M_d)\frac{d}{dM_c}\left(\frac{\Gamma(M_c+k)}{\Gamma(M_c)}\right)}{\frac{d}{dB_c}\left(\frac{\Gamma(B_c+k)}{\Gamma(B_c)}\right)} \\ &= \frac{f_k(s, M_d)}{f_k(B_c, M_c)} \cdot \frac{\Psi(M_c+k) - \Psi(M_c)}{\Psi(B_c+k) - \Psi(B_c)} \\ &= \frac{\Psi(M_c+k) - \Psi(M_c)}{\Psi(B_c+k) - \Psi(B_c)}, \qquad \text{(C.32)}\end{aligned}$$

$$\begin{aligned}\frac{df_{k+1}(B_c, M_c)}{dM_c} &= \frac{\partial f_{k+1}(B_c, M_c)}{\partial B_c}\frac{dB_c}{dM_c} + \frac{\partial f_{k+1}(B_c, M_c)}{\partial M_c} \\ = \ & f_{k+1}(B_c, M_c)\left(\Psi(B_c+k+1) - \Psi(B_c)\right)\frac{\Psi(M_c+k) - \Psi(M_c)}{\Psi(B_c+k) - \Psi(B_c)} \\ & -f_{k+1}(B_c, M_c)\left(\Psi(M_c+k+1) - \Psi(M_c)\right). \qquad \text{(C.33)}\end{aligned}$$

Let $u_i = \frac{1}{M_c+i}, v_i = \frac{1}{B_c+i}, q_i = \frac{u_i}{v_i} = \frac{B_c+i}{M_c+i}$, $i = 0, 1, \ldots, k$. Apparently, $\{q_i\}$ is an increasing sequence.

$$\begin{aligned}\frac{\Psi(M_c+k+1) - \Psi(M_c)}{\Psi(B_c+k+1) - \Psi(B_c)} &= \frac{\sum_{i=0}^{k} u_i}{\sum_{i=0}^{k} v_i} = \frac{\sum_{i=0}^{k} q_i v_i}{\sum_{i=0}^{k} v_i} = \frac{q_k v_k + \sum_{i=0}^{k-1} q_i v_i}{v_k + \sum_{i=0}^{k-1} v_i} \\ &> \frac{\sum_{i=0}^{k-1} q_i v_i}{\sum_{i=0}^{k-1} v_i} = \frac{\Psi(M_c+k) - \Psi(M_c)}{\Psi(B_c+k) - \Psi(B_c)}, \qquad \text{(C.34)}\end{aligned}$$

$$\frac{df_{k+1}(B_c, M_c)}{dM_c} < 0. \qquad \text{(C.35)}$$

Therefore, $f_{k+1}(B_c, M_c)$ is a strictly decreasing function of M_c. Since $M_c < M_d$,

$$f_{k+1}(B_c, M_c) > f_{k+1}(B_c, M_c)|_{M_c=M_d} = f_{k+1}(s, M_d). \qquad \text{(C.36)}$$

C.4.2 Proof of Lemma 5.2

This lemma is also proposed in subsection 5.1.2. It states that if the weighted average of the k-th moment error rates of a set of leaves is equal to that of another leaf, and if all leaves cover an equal number of training examples, then the weighted average (with the same weights as before) of the $(k+1)$-th moment error rate of the same set of leaves is no less than the $(k+1)$-th moment of the other leaf; the equality holds if and only if all leaves have the same number of training misclassifications. Mathematically speaking, if $\sum_i p_i g_k(x_i) = g_k(X)$, $x_i > 0$, $p_i > 0$, and $\sum_i p_i = 1$, then $\sum_i p_i g_{k+1}(x_i) \geq g_{k+1}(X)$; the equality holds if and only if $x_i = X$ for all i.

Apparently $g_k(x)$ is a strictly increasing function of x when $x > 0$, and $g_k(x)$ is unbounded when $x \to \infty$. Therefore, the inverse function of $g_k(x)$ exists for $x > 0$. Define $y_i = g_k(x_i)$, $Y = g_k(X)$, and $h(y) = g_k^{-1}(y)$. Since

$$\sum_i p_i y_i = Y, \tag{C.37}$$

we have:

$$\begin{aligned}\sum_i p_i g_{k+1}(x_i) - g_{k+1}(X) &= \sum_i p_i g_k(x_i)(x_i + k) - g_{k+1}(X)(X + k) \\ &= \sum_i p_i g_k(x_i) x_i - g_{k+1}(X) X \\ &= \sum_i p_i y_i h(y_i) - Y h(Y). \end{aligned} \tag{C.38}$$

It suffices to prove the convexity of the function $yh(y)$. In fact,

$$\frac{d^2\left(yh(y)\right)}{dy^2} = 2h^{(1)}(y) + yh^{(2)}(y), \tag{C.39}$$

$$h^{(1)}(y) = \frac{dx}{dy} = \frac{1}{g_k^{(1)}(x)}, \tag{C.40}$$

$$h^{(2)}(y) = \frac{d}{dy}\left(\frac{1}{g_k^{(1)}(x)}\right) = \frac{d}{dx}\left(\frac{1}{g_k^{(1)}(x)}\right)\frac{dx}{dy} = -\frac{g_k^{(2)}(x)}{\left(g_k^{(1)}(x)\right)^3}, \tag{C.41}$$

$$g_k^{(1)}(x) = g_k(x)\left(\Psi(x+k) - \Psi(x)\right), \tag{C.42}$$

$$g_k^{(2)}(x) = g_k(x)\left(\Psi(x+k) - \Psi(x)\right)^2 + g_k(x)\left(\Psi^{(1)}(x+k) - \Psi^{(1)}(x)\right). \tag{C.43}$$

Since $\Psi(x+k) - \Psi(x) = \sum_{j=0}^{k-1}\frac{1}{x+j}$, $\Psi^{(1)}(x+k) - \Psi^{(1)}(x) < 0$. Considering $g_k(x) > 0$ and $g_k^{(1)}(x) > 0$,

$$g_k^{(2)}(x) < g_k(x)\left(\Psi(x+k) - \Psi(x)\right)^2, \tag{C.44}$$

$$\begin{aligned}
\frac{d^2\left(yh(y)\right)}{dy^2} &= \frac{2}{g_k^{(1)}(x)} - \frac{g_k(x)g_k^{(2)}(x)}{\left(g_k^{(1)}(x)\right)^3} \\
&> \frac{\left(g_k^{(1)}(x)\right)^2 - g_k(x)g_k^{(2)}(x)}{\left(g_k^{(1)}(x)\right)^3} \\
&> \frac{\left(g_k^{(1)}(x)\right)^2 - g_k(x)^2\left(\Psi(x+k) - \Psi(x)\right)^2}{\left(g_k^{(1)}(x)\right)^3} \\
&= 0.
\end{aligned} \tag{C.45}$$

Therefore, the function $yh(y)$ is convex.

C.4.3 Proof of the theorem

Since $0 < B_c < M_c < M_d$ for any leaf c under the decision node d, and $f_k(B, M)$ is strictly increasing with respect to B and decreasing with respect to M,

$$f_k(0, M_d) \leq f_k(0, M_c) < f_k(B_c, M_c) < f_k(M_c, M_c) = f_k(M_d, M_d). \tag{C.46}$$

Note that both the bounds $f_k(0, M_d)$ and $f_k(M_d, M_d)$ are independent of c. As a result,

$$f_k(0, M_d) < \sum_{c \in Leaves(d)} f_k(B_c, M_c) p_{c|d} < f_k(M_d, M_d). \tag{C.47}$$

Therefore, there is a unique solution s in $(0, M_d)$ for the following equation:

$$\sum_{c \in Leaves(d)} f_k(B_c, M_c) p_{c|d} = f_k(s, M_d). \tag{C.48}$$

Since $f_k(s, M_d) \geq f_k(B_d, M_d)$, we know that $s \geq B_d$ and $f_{k+1}(s, M_d) \geq f_{k+1}(B_d, M_d)$. It suffices to prove:

$$\sum_{c \in Leaves(d)} f_{k+1}(B_c, M_c) p_{c|d} > f_{k+1}(s, M_d). \tag{C.49}$$

For each leaf c below d, let s_c be the solution (which is also unique) of the following equation:

$$f_k(B_c, M_c) = f_k(s_c, M_d), \tag{C.50}$$

which means

$$\sum_{c \in Leaves(d)} f_k(s_c, M_d) p_{c|d} = f_k(s, M_d), \tag{C.51}$$

namely

$$\sum_{c \in Leaves(d)} g_k(s_c) p_{c|d} = g_k(s). \tag{C.52}$$

According to Lemma 2,

$$\sum_{c \in Leaves(d)} g_{k+1}(s_c) p_{c|d} \geq g_{k+1}(s), \tag{C.53}$$

which means

$$\sum_{c \in Leaves(d)} f_{k+1}(s_c, M_d) p_{c|d} \geq f_{k+1}(s, M_d). \tag{C.54}$$

According to Lemma 1,

$$f_{k+1}(B_c, M_c) > f_{k+1}(s_c, M_d). \tag{C.55}$$

Therefore, (C.49) is proven. The proof of this theorem is completed. ∎

C.5 Proof of Theorem 5.5

The theorem states that the k-norm of the error rate of any node is a strictly increasing function of k.

Proof For a leaf, the theorem is apparently true, because the k-norm of any random variable is an increasing function of k according to Lyapunov's inequality (see [KP00]), which can be derived easily from Jensen's inequality (see [Jen06]). In the proof of Jensen's inequality we can easily see that if the random variable is not a constant number, its k-norm is strictly increasing with k. In our application, the class probabilities at any leaf obey the Dirichlet distribution, and thus the error rate is not a deterministic number. Therefore, the k-norm error rate at any leaf is a strictly increasing function of k.

For a decision node d, the k-th moment of the error rate is computed as:

$$\begin{aligned} E\left[r_d^k\right] &= \sum_{c \in Leaves(d)} E\left[r_c^k\right] p_{c|d} \\ &= \sum_{c \in Leaves(d)} p_{c|d} \int_0^1 x^k f_c(x) dx \\ &= \int_0^1 x^k \left(\sum_{c \in Leaves(d)} p_{c|d} f_c(x) \right) dx, \end{aligned} \tag{C.56}$$

where $f_c(x)$ is the PDF of the error rate in the leaf c. Define

$$g(x) = \sum_{c \in Leaves(d)} p_{c|d} f_c(x). \tag{C.57}$$

It is easily seen that $g(x)$ qualifies as a PDF. Since $f_c(x)$ is not an impulse function for any c, (i.e., none of the error rates in the leaves has a deterministic value), $g(x)$ also represents the distribution of a non-deterministic variable. Therefore, the k-norm of the error rate in a decision node is also a strictly increasing function of k. ▮

C.6 Proof of Theorem 5.6

This theorem states that an ineffective split never decreases the 1-norm error rate. The 1-norm error rate of the node t before the split is

$$r_{leaf}(t) = \frac{b_t + (J-1)\lambda}{n_t + J\lambda}. \tag{C.58}$$

After the split,

$$r_{tree}(t) = \sum_{c \in Children(t)} r_{leaf}(c) \frac{n_c + \eta}{n_t + K_t \eta}. \tag{C.59}$$

Our goal is to prove

$$r_{tree}(t) \geq r_{leaf}(t), \tag{C.60}$$

when $\sum_{c \in Children(t)} b_c = b_t$ and $0 \leq \eta \leq J\lambda$.

Proof We first find the minimum value of the left hand side of (C.60) with respect to η.

$$\frac{\partial}{\partial \eta} \sum_c \frac{(b_c + (J-1)\lambda)(n_c + \eta)}{(n_c + J\lambda)(n_t + K_t \eta)} = \sum_c \frac{(b_c + (J-1)\lambda)(n_t - K_t n_c)}{(n_c + J\lambda)(n_t + K_t \eta)^2}. \tag{C.61}$$

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

Case $\eta = 0$: Since $n_c < n_t$ and $b_t \leq \frac{J-1}{J} n_t$,

$$\begin{aligned}
r_{tree}(t) &= \sum_{c \in Children(t)} \frac{b_c + (J-1)\lambda}{n_c + J\lambda} \frac{n_c}{n_t} \\
&= \sum_{c \in Children(t)} \left(\frac{b_c}{n_c} + \frac{\lambda\left((J-1)n_c - Jb_c\right)}{n_c(n_c + J\lambda)} \right) \frac{n_c}{n_t} \\
&= \frac{b_t}{n_t} + \frac{\lambda}{n_t} \sum_{c \in Children(t)} \frac{(J-1)n_c - Jb_c}{n_c + J\lambda} \\
&\geq \frac{b_t}{n_t} + \frac{\lambda}{n_t\left(n_t + J\lambda\right)} \sum_{c \in Children(t)} \left((J-1)n_c - Jb_c\right) \\
&= \frac{b_t}{n_t} + \frac{\lambda\left((J-1)n_t - Jb_t\right)}{n_t\left(n_t + J\lambda\right)} \\
&= \frac{b_t + (J-1)\lambda}{n_t + J\lambda} \\
&= r_{leaf}(t). \qquad \text{(C.62)}
\end{aligned}$$

Case $\eta = J\lambda$:

$$\begin{aligned}
r_{tree}(t) &= \frac{\sum_{c \in Children(t)} \left(b_c + (J-1)\lambda\right)}{n_t + K_t J\lambda} = \frac{b_t + (J-1)K_t\lambda}{n_t + K_t J\lambda}, \qquad \text{(C.63)} \\
\frac{\partial r_{tree}(t)}{\partial K_t} &= \frac{(J-1)\lambda\left(n_t + K_t J\lambda\right) - \left(b_t + (J-1)K_t\lambda\right)J\lambda}{\left(n_t + K_t J\lambda\right)^2} \\
&= \frac{\lambda\left((J-1)n_t - Jb_t\right)}{\left(n_t + K_t J\lambda\right)^2} \\
&\geq 0. \qquad \text{(C.64)}
\end{aligned}$$

Since $K_t > 1$, $r_{tree}(t) \geq r_{tree}(t)|_{K_t=1} = r_{leaf}(t)$.

For both cases $\eta = 0$ and $\eta = J\lambda$, the equality holds if and only if $(J-1)n_c = Jb_c$ for all leaves, i.e., all classes have exactly the same number of examples at each leaf. ∎

C.7 Proof of Theorem 5.8

This theorem states that when k approaches ∞, the k-norm error rate of any node approaches one, meaning that when we are infinitely pessimistic, we think the prediction of every node is definitely wrong.

Proof We first prove the theorem for a leaf by proving that the log-error approaches zero as $k \to \infty$. For any leaf c,

$$\log \|r_c\|_k = \frac{\log\Gamma\left(B_c+k\right) - \log\Gamma\left(B_c\right) + \log\Gamma\left(M_c\right) - \log\Gamma\left(M_c+k\right)}{k}. \tag{C.65}$$

According to *l'Hospital's Rule*,

$$\begin{aligned} \lim_{k\to\infty} \log\|r_c\|_k &= \lim_{k\to\infty} \frac{\partial\left(\log\Gamma\left(B_c+k\right) - \log\Gamma\left(B_c\right) + \log\Gamma\left(M_c\right) - \log\Gamma\left(M_c+k\right)\right)}{\partial k} \\ &= \lim_{k\to\infty}\left(\Psi\left(B_c+k\right) - \Psi\left(M_c+k\right)\right), \end{aligned} \tag{C.66}$$

where $\Psi()$ is the digamma function that can be represented below:

$$\Psi(x) = \int_0^\infty \left(\frac{e^{-y}}{y} - \frac{e^{-xy}}{1-e^{-y}}\right) dy. \tag{C.67}$$

Apparently, when $x > 0$, $\Psi(x)$ is an increasing function of x. Therefore,

$$\Psi\left(B_c+k\right) - \Psi\left(B_c + \lceil M_c - B_c\rceil + k\right) \le \Psi\left(B_c+k\right) - \Psi\left(M_c+k\right) \le 0, \tag{C.68}$$

where $\lceil M_c - B_c \rceil$ is the ceiling function of $M_c - B_c$ and is designated by Δ_c. Since $M_c - B_c > 0$, Δ_c is a natural number. According to the property of the digamma function,

$$\Psi\left(B_c + k\right) - \Psi\left(B_c + \Delta_c + k\right) = -\sum_{i=0}^{\Delta_c - 1} \frac{1}{B_c + k + i}. \tag{C.69}$$

Note that the number of terms in the above summation is at least one and is invariant to k. Each term approaches zero as $k \to \infty$, which means

$$\lim_{k\to\infty} \left(\Psi\left(B_c + k\right) - \Psi\left(B_c + \Delta_c + k\right)\right) = 0. \tag{C.70}$$

According to the Squeezing Theorem, the k-norm of the error rate at a leaf approaches one.

Now consider a decision node d. Its k-norm of the error rate is

$$\|r_d\|_k = \left(\sum_{c\in Leaves(d)} p_{c|d} \|r_c\|_k^k\right)^{1/k}. \tag{C.71}$$

Apparently,

$$\min_{c\in Leaves(d)} \|r_c\|_k \le \|r_d\|_k \le \max_{c\in Leaves(d)} \|r_c\|_k. \tag{C.72}$$

Since $\|r_c\|_k \to 1$ for any leaf c when $k \to \infty$, $\min_c \|r_c\|_k$ and $\max_c \|r_c\|_k$ also approach 1, and so does $\|r_d\|_k$. ∎

C.8 Proof of Theorem 5.9

Theorem 5.9 provides a necessary and sufficient condition for a sub-tree to be beneficial under any k-norm error rate criterion for finite k. We use the same notation as in the proof of Theorem 5.3. This theorem is re-stated below: $\sum_{c \in Leaves(d)} f_k(B_c, M_c) p_{c|d} < f_k(B_d, M_d)$ *for all natural numbers k if and only if $M_c - B_c = M_d - B_d$ for all leaves c below d.*

Proof We first prove the sufficiency and then prove the necessity.

C.8.1 Sufficiency

If $M_c - B_c = M_d - B_d$ for all leaves c below d, we have the following inequality for any non-negative number i (since $M_c < M_d$):

$$\frac{B_c + i}{M_c + i} = 1 - \frac{M_c - B_c}{M_c + i} < 1 - \frac{M_d - B_d}{M_d + i} = \frac{B_d + i}{M_d + i}. \tag{C.73}$$

Therefore, for any natural number k,

$$f_k(B_c, M_c) = \prod_{i=0}^{k-1} \frac{B_c + i}{M_c + i} < \prod_{i=0}^{k-1} \frac{B_d + i}{M_d + i} = f_k(B_d, M_d), \tag{C.74}$$

which implies that

$$\sum_{c \in Leaves(d)} f_k(B_c, M_c) p_{c|d} < f_k(B_d, M_d). \tag{C.75}$$

C.8.2 Necessity

Assume there exists a leaf s below d such that $M_s - B_s \neq M_d - B_d$. To prove that there exists a threshold k_2 such that for any $k > k_2$, $\sum_{c \in Leaves(d)} f_k(B_c, M_c) p_{c|d} > f_k(B_d, M_d)$, it suffices to prove that

$$\lim_{k\to\infty} \frac{f_k(B_s, M_s) p_{s|d}}{f_k(B_d, M_d)} = p_{s|d} \lim_{k\to\infty} \prod_{i=0}^{k-1} \frac{(M_d + i)(B_s + i)}{(M_s + i)(B_d + i)} = \infty. \tag{C.76}$$

Since $n_{j,s} \leq n_{j,d}$ for any j,

$$M_s - B_s = \max_j n_{j,s} + (J-1)\lambda \leq \max_j n_{j,d} + (J-1)\lambda = M_d - B_d. \tag{C.77}$$

According to our assumption, $M_s - B_s < M_d - B_d$. In order to demonstrate (C.76), we first prove that there exists $\varepsilon_1 > 0$ and $k_1 > 0$ such that for any $k > k_1$,

$$\frac{(M_d + k)(B_s + k)}{(M_s + k)(B_d + k)} > \frac{(k+1)^{\varepsilon_1}}{k^{\varepsilon_1}}. \tag{C.78}$$

Let $\varepsilon = M_d + B_s - M_s - B_d > 0$,

$$\frac{(M_d + k)(B_s + k)}{(M_s + k)(B_d + k)} = 1 + \frac{\varepsilon k + M_d B_s - M_s B_d}{(M_s + k)(M_d + k)}. \tag{C.79}$$

There exist ε_1 and ε_2 such that $0 < \varepsilon_1 < \varepsilon_2 < \varepsilon$. Since

$$\lim_{k\to\infty} \frac{(\varepsilon k + M_d B_s - M_s B_d)\, k}{(M_s + k)\,(M_d + k)\,\varepsilon_2} = \frac{\varepsilon}{\varepsilon_2} > 1, \tag{C.80}$$

and according to *l'Hospital's Rule*,

$$\lim_{k\to\infty} \frac{\log\left(1 + \frac{\varepsilon_2}{k}\right)}{\log \frac{(k+1)^{\varepsilon_1}}{k^{\varepsilon_1}}} = \lim_{k\to\infty} \frac{\frac{d}{dk}\log\left(1 + \frac{\varepsilon_2}{k}\right)}{\varepsilon_1 \frac{d}{dk}\log\frac{k+1}{k}} = \lim_{k\to\infty} \frac{\frac{\varepsilon_2}{k(k+\varepsilon_2)}}{\frac{\varepsilon_1}{k(k+1)}} = \frac{\varepsilon_2}{\varepsilon_1} > 1, \tag{C.81}$$

when k is above a certain threshold k_1,

$$\frac{(M_d + k)\,(B_s + k)}{(M_s + k)\,(B_d + k)} = 1 + \frac{\varepsilon k + M_d B_s - M_s B_d}{(M_s + k)\,(M_d + k)} > 1 + \frac{\varepsilon_2}{k} > \frac{(k+1)^{\varepsilon_1}}{k^{\varepsilon_1}}, \tag{C.82}$$

which means (C.78) is true. Therefore,

$$\lim_{k\to\infty} \frac{f_k(B_s, M_s) p_{s|d}}{f_k(B_d, M_d)} = p_{s|d} \lim_{k\to\infty} \prod_{i=0}^{k-1} \frac{(M_d + i)\,(B_s + i)}{(M_s + i)\,(B_d + i)} = \alpha \lim_{k\to\infty} \prod_{i=k_1+1}^{k-1} \frac{(M_d + i)\,(B_s + i)}{(M_s + i)\,(B_d + i)}, \tag{C.83}$$

where α is positive and independent of k:

$$\alpha = p_{s|d} \prod_{i=0}^{k_1} \frac{(M_d + i)\,(B_s + i)}{(M_s + i)\,(B_d + i)} > 0. \tag{C.84}$$

According to (C.78),

$$\lim_{k\to\infty} \frac{f_k(B_s, M_s)p_{s|d}}{f_k(B_d, M_d)} \geq \alpha \lim_{k\to\infty} \prod_{i=k_1+1}^{k-1} \frac{(i+1)^{\varepsilon_1}}{i^{\varepsilon_1}} = \alpha \lim_{k\to\infty} \frac{k^{\varepsilon_1}}{(k_1+1)^{\varepsilon_1}} = \infty. \tag{C.85}$$

Therefore, there exists a threshold k_2 such that $\sum_{c\in Leaves(d)} f_k(B_c, M_c)p_{c|d} > f_k(B_d, M_d)$ for any $k > k_2$. ∎

C.9 Proof of Equation (7.14)

Define

$$\begin{aligned} q_j &= R_{Label(l),j}, & \text{(C.86)} \\ p_j &= P\left[C_j|A_l\right], & \text{(C.87)} \\ M_j &= n_{j,l} + \lambda & \text{(C.88)} \\ M &= \sum_{j=1}^{J} M_j & \text{(C.89)} \end{aligned}$$

then

$$\begin{aligned} E_k^* &= E\left[\left(\sum_{j=1}^{J} q_j p_j\right)^k\right] \\ &= E\left[\sum_{k_1+k_2+\ldots+k_J=k} \frac{k!}{k_1!\ldots k_J!}\left(q_1^{k_1}\ldots q_J^{k_J}\right)\left(p_1^{k_1}\ldots p_J^{k_J}\right)\right], & \text{(C.90)} \end{aligned}$$

Define

$$g_n(x) = \frac{\Gamma(x+n)}{\Gamma(x)}, \tag{C.91}$$

where n is a non-negative integer. According to (C.4),

$$E\left[p_1^{n_1} \dots p_J^{n_J}\right] = \frac{g_{n_1}(M_1) \dots g_{n_J}(M_J)}{g_k(M)}. \tag{C.92}$$

For $k > 1$,

$$\begin{aligned}
E_k^* &= E\left[\left(\sum_{j=1}^{J} q_j p_j\right)^k\right] \\
&= E\left[\left(\sum_{i=1}^{J} q_i p_i\right)\left(\sum_{j=1}^{J} q_j p_j\right)^{k-1}\right] \\
&= E\left[\sum_{i=1}^{J}\left(\sum_{j=1}^{J} q_j p_j\right)^{k-1} q_i p_i\right] \\
&= E\left[\sum_{i=1}^{J} \sum_{k_1+\ldots+k_J=k-1} \frac{(k-1)!}{k_1! \dots k_J!}\left(q_1^{k_1} \dots q_J^{k_J}\right)\left(p_1^{k_1} \dots p_J^{k_J}\right) q_i p_i\right] \\
&= \sum_{i=1}^{J} \sum_{k_1+\ldots+k_J=k-1} \frac{(k-1)!}{k_1! \dots k_J!}\left(q_1^{k_1} \dots q_J^{k_J}\right) \\
&\quad \cdot E\left[p_1^{k_1} \dots p_{(i-1)}^{k_{(i-1)}} p_i^{(k_i)+1} p_{(i+1)}^{k_{(i+1)}} \dots p_J^{k_J}\right] q_i \\
&= \sum_{i=1}^{J} \sum_{k_1+\ldots+k_J=k-1} \frac{(k-1)!}{k_1! \dots k_J!}\left(q_1^{k_1} \dots q_J^{k_J}\right) \\
&\quad \cdot \frac{g_{k_1}(M_1) \dots g_{k_{(i-1)}}(M_{(i-1)}) g_{(k_i)+1}(M_i) g_{k_{(i+1)}}(M_{(i+1)}) \dots g_{k_J}(M_J)}{g_k(M)} q_i
\end{aligned} \tag{C.93}$$

According to the property of Gamma function,

$$g_n(x) = \prod_{m=0}^{n-1}(x+m), \tag{C.94}$$

and therefore, for $k > 1$,

$$g_n(x) = g_{(n-1)}(x)(x+n-1). \tag{C.95}$$

Thus,

$$\begin{aligned}
E_k^* &= \sum_{i=1}^{J} \sum_{k_1+\ldots+k_J=k-1} \frac{(k-1)!}{k_1!\ldots k_J!}\left(q_1^{k_1}\ldots q_J^{k_J}\right)\frac{g_{k_1}(M_1)\ldots g_{k_J}(M_J)(M_i+k_i)q_i}{g_{k-1}(M)(M+k-1)} && \text{(C.96)}\\
&= \sum_{i=1}^{J} \sum_{k_1+\ldots+k_J=k-1} \frac{(k-1)!}{k_1!\ldots k_J!}\left(q_1^{k_1}\ldots q_J^{k_J}\right)\frac{g_{k_1}(M_1)\ldots g_{k_J}(M_J)M_iq_i}{g_{k-1}(M)(M+k-1)}\\
&\quad +\sum_{i=1}^{J} \sum_{k_1+\ldots+k_J=k-1} \frac{(k-1)!}{k_1!\ldots k_J!}\left(q_1^{k_1}\ldots q_J^{k_J}\right)q_i\frac{g_{k_1}(M_1)\ldots g_{k_J}(M_J)k_i}{g_{k-1}(M)(M+k-1)} && \text{(C.97)}
\end{aligned}$$

Using (C.92) again,

$$\begin{aligned}
E_k^* &= E_{k-1}^*\frac{\sum_{i=1}^{J} M_iq_i}{M+k-1} + \sum_{i=1}^{J} \sum_{k_1+\ldots+k_J=k-1,k_i>0} \frac{(k-1)!}{k_1!\ldots k_{(i-1)}!((k_i)-1)!k_{(i+1)}!\ldots k_J!}\\
&\quad \cdot\left(q_1^{k_1}\ldots q_J^{k_J}\right)q_i\frac{g_{k_1}(M_1)\ldots g_{k_J}(M_J)}{g_{k-1}(M)(M+k-1)} && \text{(C.98)}
\end{aligned}$$

Let $k'_j = k_j$ for $j \neq i$ and let $k'_i = k_i - 1$. Then,

$$\begin{aligned}
E^*_k &= E^*_{k-1}\frac{\sum_{i=1}^J M_i q_i}{M+k-1} + \sum_{i=1}^J \sum_{k'_1+\ldots+k'_J=k-2} \frac{(k-1)!}{k'_1!\ldots k'_J!}\left(q_1^{k'_1}\ldots q_J^{k'_J}\right) q_i^2 \\
&\cdot \frac{g_{k'_1}(M_1)\ldots g_{k'_{(i-1)}}(M_{(i-1)}) g_{(k'_i)+1}(M_i) g_{k'_{(i+1)}}(M_{(i+1)})\ldots g'_{k_J}(M_J)}{g_{k-1}(M)(M+k-1)} \\
&= E^*_{k-1}\frac{\sum_{i=1}^J M_i q_i}{M+k-1} + \sum_{i=1}^J \sum_{k'_1+\ldots+k'_J=k-2} \frac{(k-1)(k-2)!}{k'_1!\ldots k'_J!} \\
&\cdot \left(q_1^{k'_1}\ldots q_J^{k'_J}\right) \frac{g_{k'_1}(M_1)\ldots g_{k'_J}(M_J)(M_i+k'_i)q_i^2}{g_{k-2}(M)(M+k-1)(M+k-2)} \\
&= E^*_{k-1}\frac{\sum_{i=1}^J M_i q_i}{M+k-1} + \frac{k-1}{M+k-1}\sum_{i=1}^J \sum_{k'_1+\ldots+k'_J=k-2} \frac{(k-2)!}{k'_1!\ldots k'_J!} \\
&\cdot \left(q_1^{k'_1}\ldots q_J^{k'_J}\right) \frac{g_{k'_1}(M_1)\ldots g_{k'_J}(M_J)(M_i+k'_i)q_i^2}{g_{k-2}(M)(M+k-2)} \qquad \text{(C.99)}
\end{aligned}$$

Note that the second term in the right hand side of the above equation has exactly the same form as (C.96), and thus the above process can be repeated, yielding

$$E^*_k = \sum_{m=1}^k \frac{\frac{(k-1)!}{(k-m)!} E^*_{k-m} \sum_{i=1}^J M_i q_i^m}{\prod_{i=1}^m (M+k-m)} \qquad \text{(C.100)}$$

∎

LIST OF REFERENCES

[BB94] Marko Bohanec and Ivan Bratko. "Trading Accuracy for Simplicity in Decision Trees." *Machine Learning*, **15**(3):223–250, 1994.

[BFO84] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees.* Wadsworth, 1984.

[BKK98] Jeffrey P. Bradford, Clayton Kunz, Ron Kohavi, Clifford Brunk, and Carla E. Brodley. "Pruning Decision Trees with Misclassification Costs." In Claire Nedellec and Céline Rouveirol, editors, *ECML*, volume 1398 of *Lecture Notes in Computer Science*, pp. 131–136. Springer, 1998.

[BN03] N. Balakrishnan and V. B. Nevzorov. *A Primer on Statistical Distributions.* Wiley, Hoboken, N.J., 2003.

[Bra97] Andrew P. Bradley. "The use of the area under the ROC curve in the evaluation of machine learning algorithms." *Pattern Recognition*, **30**(7):1145–1159, 1997.

[Bre01] Leo Breiman. "Random Forests." *Machine Learning*, **45**(1):5–32, 2001.

[CB91] Bojan Cestnik and Ivan Bratko. "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, NY, USA, 1991. Springer-Verlag New York, Inc.

[Ces90] Bojan Cestnik. "Estimating Probabilities: A Crucial Task in Machine Learning." In *ECAI*, pp. 147–149, 1990.

[CGM92] Gail A. Carpenter, Stephen Grossberg, Natalya Markuzon, John H. Reynolds, and David B. Rosen. "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps." *IEEE Transactions on Neural Networks*, **3**(5):698–713, September 1992.

[Che52] Herman Chernoff. "A measure of asymptotic efficiency of tests of a hypothesis based on the sum of observations." *Annals of Mathematical Statistics*, **23**:493–507, 1952.

[CKG01] Dimitrios Charalampidis, Takis Kasparis, and Michael Georgiopoulos. "Classification of Noisy Signals Using Fuzzy ARTMAP Neural Networks." *IEEE Transactions on Neural Networks*, **12**(5):1023–1036, September 2001.

[DHS00] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, November 2000.

[Die00] Thomas G. Dietterich. “An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization.” *Machine Learning*, **40**(2):139–157, 2000.

[DK01] Ming Dong and Ravi Kothari. “Classifiability Based Pruning of Decision Trees.” In *IJCNN*, volume 3, pp. 1739–1743, 2001.

[EK01] Tapio Elomaa and Matti Kääriäinen. “An analysis of reduced error pruning.” *Journal of Articial Intelligence Research*, **15**:163–187, 2001.

[EMS97] Floriana Esposito, Donato Malerba, and Giovanni Semeraro. “A Comparative Analysis of Methods for Pruning Decision Trees.” *IEEE Trans. Pattern Anal. Mach. Intell.*, **19**(5):476–491, 1997.

[FFH03] César Ferri, Peter A. Flach, and José Hernández-Orallo. “Improving the AUC of Probabilistic Estimation Trees.” In Nada Lavrac, Dragan Gamberger, Ljupco Todorovski, and Hendrik Blockeel, editors, *ECML*, volume 2837 of *Lecture Notes in Computer Science*, pp. 121–132. Springer, 2003.

[Fre98] Yoav Freund. “Self Bounding Learning Algorithms.” In *COLT: Proceedings of the Workshop on Computational Learning Theory*. Morgan Kaufmann Publishers Inc., 1998.

[GCS95] A. Gelman, J.B. Carlin, H.S. Stern, and D.B. Rubin. *Bayesian Data Analysis*. Chapman and Hall, London, 1995.

[Goo65] Irving John Good. *The Estimation of Probabilities: An Essay on Modern Bayesian Methods*. Number 30 in Research monographs. MIT Press, Cambridge, MA, 1965.

[Goo67] Irving John Good. “A Bayesian Significance Test for Multinomial Distributions.” *Journal of the Royal Statistical Society. Series B (Methodological)*, **29**(3):399–431, 1967.

[HBB03] Lawrence O. Hall, Kevin W. Bowyer, Robert E. Banfield, Steven Eschrich, and Richard Collins. “Is Error-Based Pruning Redeemable?” *International Journal on Artificial Intelligence Tools*, **12**(3):249–264, 2003.

[Hoe63] Wassily Hoeffding. “Probability inequalities for sums of bounded random variables.” *Journal of the American Statistical Association*, **58**:13–30, 1963.

[HR76] Laurent Hyafil and Ronald L. Rivest. “Constructing Optimal Binary Decision Trees is NP-Complete.” *Inf. Process. Lett.*, **5**(1):15–17, 1976.

[Jen06] J.L.W.V. Jensen. "Sur les fonctions convexes et les inégalités entre les valeurs moyennes." *Acta Math*, **30**:175–193, 1906.

[KBS97] Ron Kohavi, Barry Becker, and Dan Sommerfield. "Improving Simple Bayes." In Maarten van Someren and Gerhard Widmer, editors, *ECML*, volume 1224 of *Lecture Notes in Computer Science*, pp. 78–87. Springer, 1997.

[KC01] Boonserm Kijsirikul and Kongsak Chongkasemwongse. "Decision tree pruning using backpropagation neural networks." In *IJCNN*, volume 3, pp. 1876–1880, 2001.

[KE03] Matti Kääriäinen and Tapio Elomaa. "Rademacher Penalization over Decision Tree Prunings." In Nada Lavrac, Dragan Gamberger, Ljupco Todorovski, and Hendrik Blockeel, editors, *ECML*, volume 2837 of *Lecture Notes in Computer Science*, pp. 193–204. Springer, 2003.

[KM98] Michael J. Kearns and Yishay Mansour. "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, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[KME04] Matti Kääriäinen, Tuomo Malinen, and Tapio Elomaa. "Selective Rademacher Penalization and Reduced Error Pruning of Decision Trees." *Journal of Machine Learning Research*, **5**:1107–1126, 2004.

[KP00] Peter E. Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations (Stochastic Modelling and Applied Probability).* Springer, November 2000.

[Kri98] Rafail E. Krichevskiy. "Laplace's Law of Succession and Universal Encoding." *IEEE Transactions on Information Theory*, **44**(1):296–303, 1998.

[Lid20] G. J. Lidstone. "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, 1920.

[LLS00] Tjen-Sien Lim, Wei-Yin Loh, and Yu-Shan Shih. "A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-Three Old and New Classification Algorithms." *Machine Learning*, **40**(3):203–228, 2000.

[Man97] Yishay Mansour. "Pessimistic decision tree pruning based on tree size." In *Proc. 14th International Conference on Machine Learning*, pp. 195–201. Morgan Kaufmann, 1997.

[MH95] Shaun Marriott and Robert F. Harrison. "A modified fuzzy ARTMAP architecture for the approximation of noisy mappings." *Neural Networks*, **8**(4):619–641, 1995.

[MM00] Yishay Mansour and David A. McAllester. "Generalization Bounds for Decision Trees." In *COLT '00: Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pp. 69–74, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[MRA95] Manish Mehta, Jorma Rissanen, and Rakesh Agrawal. "MDL-Based Decision Tree Pruning." In *KDD*, pp. 216–221, 1995.

[NB86] Tim Niblett and Ivan Bratko. "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, 1986.

[NHB98] D. J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. "UCI Repository of machine learning databases.", 1998.

[PD03] Foster J. Provost and Pedro Domingos. "Tree Induction for Probability-Based Ranking." *Machine Learning*, **52**(3):199–215, 2003.

[PMM94] Michael J. Pazzani, Christopher J. Merz, Patrick M. Murphy, Kamal Ali, Timothy Hume, and Clifford Brunk. "Reducing Misclassification Costs." In *ICML*, pp. 217–225, 1994.

[Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[Qui99] J. Ross Quinlan. "Simplifying decision trees." *Int. J. Hum.-Comput. Stud.*, **51**(2):497–510, 1999.

[Ris95] Eric Sven Ristad. "A Natural Law of Succession." Technical Report TR-495-95, Princeton University, 1995.

[SGF95] Padhraic Smyth, Alexander Gray, and Usama M. Fayyad. "Retrofitting Decision Tree Classifiers Using Kernel Density Estimation." In *ICML*, pp. 506–514, 1995.

[Sto78] M. Stone. "Cross-validation: A review." *Mathematics, Operations and Statistics*, **9**:127–140, 1978.

[VJ01] Stephen B. Vardeman and John Marcus Jobe. *Basic Engineering Data Collection and Analysis.* Brooks/Cole Thompson Learning, 2001.

[WA01] Terry Windeatt and Gholamreza Ardeshir. "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, UK, 2001. Springer-Verlag.

[Zem87] A.H. Zemanian. *Distribution Theory and Transform Analysis: An Introduction to Generalized Functions, with Applications.* Dover, New York, U.S.A., 1987.