Department of Computer Science

Hamilton, New Zealand

Pruning Decision Trees and Lists

A thesis

submitted in partial fulfilment

of the requirements for the degree

of

Doctor of Philosophy

at the

University of Waikato

by

Eibe Frank

January 2000

Abstract

Machine learning algorithms are techniques that automatically build models describing the structure at the heart of a set of data. Ideally, such models can be used to predict properties of future data points and people can use them to analyze the domain from which the data originates.

Decision trees and lists are potentially powerful predictors and embody an explicit representation of the structure in a dataset. Their accuracy and comprehensibility depends on how concisely the learning algorithm can summarize this structure. The final model should not incorporate spurious effects—patterns that are not genuine features of the underlying domain. Given an efficient mechanism for determining when a particular effect is due to chance alone, non-predictive parts of a model can be eliminated or "pruned." Pruning mechanisms require a sensitive instrument that uses the data to detect whether there is a genuine relationship between the components of a model and the domain. Statistical significance tests are theoretically well-founded tools for doing exactly that.

This thesis presents pruning algorithms for decision trees and lists that are based on significance tests. We explain why pruning is often necessary to obtain small and accurate models and show that the performance of standard pruning algorithms can be improved by taking the statistical significance of observations into account. We compare the effect of parametric and non-parametric tests, analyze why current pruning algorithms for decision lists often prune too aggressively, and review related work—in particular existing approaches that use significance tests in the context of pruning. The main outcome of this investigation is a set of simple pruning algorithms that should prove useful in practical data mining applications.

Acknowledgments

This thesis was written in a very supportive and inspiring environment, and I would like to thank all the people in the Department of Computer Science at the University of Waikato, especially the members of the Machine Learning Group, for making it so much fun to do my research here in New Zealand.

Most of all I would like to thank my supervisor Ian Witten for his generous support, for the freedom that allowed me to pursue several interesting little projects during the course of my PhD, for his insightful comments on all aspects of my work, and for his persistent attempts at teaching me how to write English prose.

Many thanks also to current and former members of the WEKA Machine Learning Group, in particular Geoff Holmes, Mark Hall, Len Trigg, and Stuart Inglis, and to my fellow PhD students Gordon Paynter and Yong Wang, for being such a great team to work with.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The widespread use of computers in today's society means that large quantities of data are stored electronically. This data relates to virtually all facets of modern life and is a valuable resource if the right tools are available for putting it to use. Machine learning algorithms are a set of techniques that automatically build models describing the structure at the heart of a set of data.

Such models have two important applications. First, if they accurately represent the structure underlying the data, they can be used to predict properties of future data points. Second, if they summarize the essential information in human-readable form, people can use them to analyze the domain from which the data originates.

These two applications are not mutually exclusive. To be useful for analysis, a model must be an accurate representation of the domain, and that makes it useful for prediction as well. However, the reverse is not necessarily true: some models are designed exclusively for prediction and do not lend themselves naturally to analysis. In many practical data mining problems this "black box" approach is a serious drawback because users cannot determine how a prediction is derived and match this information with their knowledge of the domain. This makes it impossible to use these models in critical applications in which a domain expert must be able to verify the decision process that leads to a prediction—for example, in medical applications.

Decision trees and lists are potentially powerful predictors that embody an explicit representation of the structure in a dataset. Moreover, compared to other sophisticated models, they can be generated very quickly. This makes them extremely useful tools for many practical data mining problems, where both predictive accuracy and the ability to analyze the model are important. The accuracy and comprehensibility of these models depends on how concisely the learning algorithm can summarize the structure in a dataset. Apart from being sufficiently accurate, the model should be as small as possible because the size of a model directly affects how easy it is to interpret. It is particularly important that it does not describe spurious effects—patterns that are not genuine features of the underlying domain.

Spurious patterns occur because datasets consist of a finite number of randomly sampled observations from an unknown probability distribution, and the stochastic nature of the sampling process does not guarantee that every observed pattern reflects structure in the underlying domain. Patterns can be due solely to noise. Learning algorithms will include these patterns in the model, unless appropriate precautions are taken. This makes the model harder to understand and often reduces the accuracy of future predictions. To avoid superfluous complexity, an efficient mechanism is needed for determining when a particular effect is due to chance alone. Given such a mechanism, those parts of a model that describe chance effects can be eliminated.

The process of cutting off non-predictive parts of a model is called "pruning." According to Webster's dictionary (1999), to prune means, among other things, "to remove as superfluous." By removing superfluous structure, pruning mechanisms reduce the size of a model and often improve its accuracy. They form an essential component of practical learning algorithms because many real-world datasets are noisy—they contain a degree of uncertainty that is either inherent in the domain, or results from the way in which the data is collected, or both.

The pruning mechanism's efficiency determines the size and accuracy of the final model. Ideally, pruning should only discard those parts of a model that are due to noise, and never eliminate any structure that is truly predictive. This decision must be based on the data at hand. Thus pruning mechanisms require a sensitive instrument that uses the given observations to detect whether there is a genuine relationship between the components of a model and the domain.

Statistical significance tests are theoretically well-founded tools for doing exactly that. They base the significance of a relationship on the probability that the observed association is due to chance alone. If this probability is low, the component is likely to describe a genuine effect; otherwise it can probably be discarded without degrading performance. Consequently significance tests are promising candidates for the task of deciding when to prune.

This thesis presents improved pruning algorithms for decision trees and lists that use significance tests as the main instrument for pruning. We explain why pruning is often necessary to obtain small and accurate models and show that the performance of standard pruning algorithms can be improved by taking the statistical significance of observations into account. We compare the effect of parametric and non-parametric tests, analyze why current pruning algorithms for decision lists often prune too aggressively, and review related work—in particular existing approaches that use significance tests in the context of pruning.

The remainder of this chapter is organized as follows. Section 1.1 introduces the basic concepts of machine learning that are used throughout this thesis. Section 1.2 explains what decision trees and lists are. Section 1.3 states the main argument of the thesis and includes an outline of the thesis' structure showing how this argument will be substantiated. Section 1.4 details the experimental methodology employed. Section 1.5 contains some historical remarks, and briefly reviews machine learning research conducted by the author that is described in other publications.

1.1 Basic Concepts

Inductive learning algorithms take some data collected from a domain as input and produce a model of the domain's structure as output. In other words, they induce a model of the domain from a given set of observations. The individual observations are called "instances," and a set of observations is called a "dataset." Each instance contains a set of values that measure certain properties of the instance. These properties are called "attributes." Each instance is described by the same set of attributes. Most implementations of learning algorithms assume that the attributes are either "nominal" or "numeric." Nominal attributes consist of a set of unordered values, for example, a set of colors. Numeric attributes can be either integers or real numbers. There are several other possible attribute types (Witten & Frank, 2000), but it is generally straightforward to adapt existing learning algorithms to deal with them. The space of all possible combinations of attribute values is called the "instance space."

Decision trees and lists belong to a group of models called "classifiers." Classifiers divide the instance space into disjoint regions and assign one of a fixed set of unordered values to each region. In other words, they assume that each instance in the instance space is labeled with an additional nominal attribute value, called the "class" of the instance. Each region of instance space is assigned to exactly one class, but more than one region can be assigned to the same class. In other words, classifiers define a function that maps the instance space onto a set of unordered values. They differ from methods for numeric prediction—for example, linear regression (Wild & Weber, 1995)—because the target value is nominal rather than numeric. Thus they can be used to model domains that pose prediction problems that have a nominal target value. These problems are called "classification problems".

Learning algorithms for classification problems have many practical applications. Consider, for example, one of the first fielded applications of classification learning: the diagnosis of soybean diseases (Michalski & Chilausky, 1980). In this application, the individual instances are soybean plants that are described by a set of attributes. Most of the attributes correspond to symptoms of various soybean diseases and their values indicate whether a particular symptom is present or absent. The class values are the different soybean diseases that can occur. A classifier for this problem defines a function that maps a particular combination of attribute values to a corresponding disease. This means that the classifier can be used to automatically obtain a diagnosis for a particular soybean plant, given a set of observed attribute values.

The task of a learning algorithm is to induce a classifier automatically from a set of "training" instances. These instances have been randomly collected from the domain and have class labels assigned to them by some other process—for example, by a human expert for soybean diseases. The learning algorithm constructs a classifier by partitioning the instance space according to the class labels of the training instances. Ideally the induced classifier will maximize the number of correctly assigned class values for all possible instances—even instances that have not occurred in the training data. In that case the learning algorithm has correctly identified the structure of the domain.

If an instance is assigned to the wrong class, we say that it is "misclassified." The predictive performance of a classifier is measured by its "error rate": the expected percentage of misclassifications on independent instances randomly sampled from the domain. Given that the structure of the domain is unknown, an infinite number of instances is required to obtain the true value of the error rate. In practice, infinite amounts of labeled data are not available, and the error rate must be estimated using an independent set of labeled instances that are unavailable to the learning algorithm when it generates the classifier. This set of instances is called the "test" data. Unlike the error rate on the training data, the observed error on the test data is an unbiased estimate of the classifier's error rate on future instances.

In many practical classification problems no learning algorithm can achieve an error rate of zero, even given an infinite amount of training data. This is because most domains contain a certain amount of "noise." If noise is present, there is a non-zero probability that different class labels will be observed if the same instance is sampled multiple times from the domain. There are several possible reasons for the occurrence of noise, for example, errors in measuring the attribute and class values of an instance. There can also be a degree of uncertainty inherent in the domain—for example, uncertainty due to the fact that not all relevant properties of an instance are known.

Apart from affecting the minimum error rate that can be achieved—a consequence that cannot be avoided by improving the learning algorithm—noise also has a detrimental effect because it potentially misleads the learning algorithm when the classifier is induced. This phenomenon can further decrease the classifier's performance. The learning algorithm can be misled by noise because training instances may have an "incorrect" class label assigned to them—a class label different from the the one that is most likely to be assigned to the same instance in the test data. This is a problem because the learning algorithm constructs a classifier according to the class labels from the training data. Consequently it is important to detect instances that are labeled incorrectly and prevent them from affecting the structure of the classifier. If the classifier fits the training instances too closely, it may fit noisy instances, and that reduces its usefulness.

This phenomenon is called "overfitting," and various heuristics have been developed to deal with it. In decision trees and lists, a common strategy is to eliminate

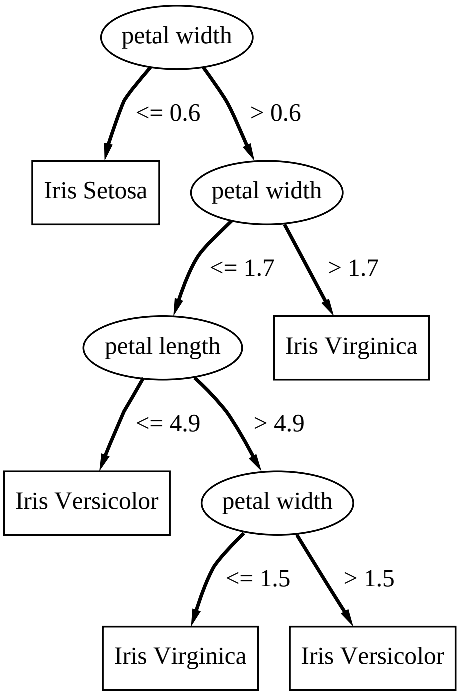


Figure 1.1: A decision tree for classifying Iris flowers

those parts of a classifier that are likely to overfit the training data. This process is called "pruning," and can increase both the accuracy and the comprehensibility of the resulting classifier. The success of a pruning mechanism depends on its ability to distinguish noisy instances from predictive patterns in the training data.

1.2 Decision Trees and Lists

Decision trees (Quinlan, 1986b) and lists (Rivest, 1987) are two closely related types of classifier. In contrast to most other classification paradigms, for example, instance-based learning (Aha et al., 1991), neural networks (Rumelhart & McClelland, 1986), Bayesian networks (Jordan, 1999), and logistic regression (Agresti, 1990), they embody an explicit representation of all the knowledge that has been induced from the training data. Given a standard decision tree or list, a user can determine manually how a particular prediction is derived, and which attributes are relevant in the derivation—without performing any numeric operations (other than comparison). This makes it very easy to explain how these classifiers are to be interpreted, and how they generate a prediction.

Figure 1.1 shows a decision tree for classifying Iris flowers according to their petal length and width. To derive a prediction, a test instance is filtered down the tree, starting from the root node, until it reaches a leaf. At each node one of the instance's attributes is tested, and the instance is propagated to the branch that corresponds to the outcome of the test. The prediction is the class label that is attached to the leaf. Note that, although each node in Figure 1.1 has a binary outcome, decision trees often exhibit nodes with more than two children. The tree from Figure 1.1 is called "univariate" because only one attribute is involved in the decision at each node. "Multivariate" decision trees can test for higher-order relationships that involve more than one attribute, for example, linear combinations of attribute values (Brodley & Utgoff, 1995). This makes them potentially more powerful predictors. However, they are also harder to interpret and computationally more expensive to generate.

Standard learning algorithms for decision trees, for example, C4.5 (Quinlan, 1992) and CART (Breiman et al., 1984), generate a tree structure by splitting the training data into smaller and smaller subsets in a recursive top-down fashion. Starting with all the training data at the root node, at each node they choose a split and divide the training data into subsets accordingly. They proceed recursively by partitioning each of the subsets further. Splitting continues until all subsets are "pure," or until their purity cannot be increased any further. A subset is pure if it contains instances of only one class. The aim is to achieve this using as few splits as possible so that the resulting decision tree is small and the number of instances supporting each subset is large. To this end, various split selection criteria have been designed, for example the information gain (Quinlan, 1986b), the Gini index (Breiman et al., 1984), and the gain ratio (Quinlan, 1992). They all provide ways of measuring the purity of a split. Some also consider the resulting support in each subset. At each node, the learning algorithm selects the split that corresponds to the best value for the splitting criterion.

Straightforward purity-based tree induction cannot deal successfully with noisy data because the strategy of creating pure subsets will isolate incorrectly labeled instances and use them for prediction. Consequently most decision tree inducers incorporate a pruning facility to deal with noisy data by eliminating unreliable branches or subtrees. Two different regimes for pruning are used: "post-pruning" and "pre-pruning." Post-pruning is invoked after the full tree has been created, and

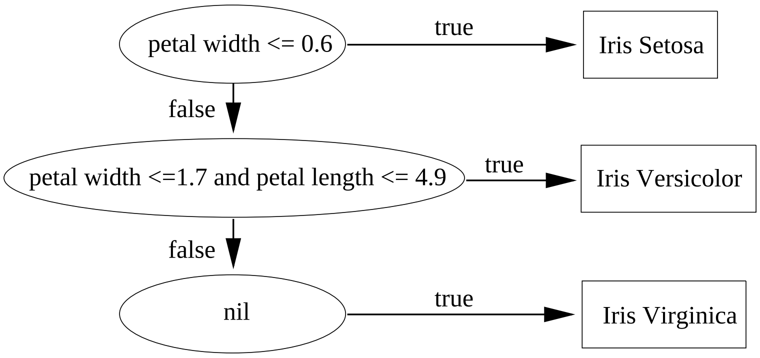


Figure 1.2: A decision list for classifying Iris flowers

deletes those parts of the classifier that do not improve its predictive performance. Algorithms for performing post-pruning will be discussed in more detail in Chapter 3. Pre-pruning, on the other hand, attempts to avoid the problem of noise by terminating the splitting process if further splits are likely to overfit the training data. This class of pruning methods will be investigated in Chapter 4.

Decision lists are similar to decision trees in that a sequence of decisions is required to derive a prediction. The difference is that all decisions have a binary outcome, true or false, and further tests on an instance are only required if the outcome of all previous decisions was negative. Individual decisions are made according to a "rule" that consists of combinations of attribute-value tests and a class label. A decision list is a list of rules that are evaluated in sequence. A rule "fires" if a test instance passes each attribute-value test that the rule contains. In that case the classification process stops and the rule's class label is assigned to the test instance. Otherwise, the next rule in the list is evaluated.

Figure 1.2 shows a decision list for the above classification problem, consisting of three rules. The last rule in the list is called the "default rule" and fires for every instance that reaches it. The default rule is required so that no instances are left unclassified. Compared to a decision tree, this rule-based classifier has the advantage that it can impose an ordering on the knowledge that is acquired: rules that are frequently used and reliable can be presented at the start of the list, and those that are less important and not very accurate deferred until the end. In some domains decision lists can represent the underlying structure of the domain much more succinctly than decision trees. The reason for this phenomenon is discussed in

Chapter 5.

The way in which decision lists are generally created is quite similar to the standard learning procedure for decision trees. The top-down induction of decision trees proceeds according to a "divide-and-conquer" strategy where the training data is partitioned into subsets and the algorithm is applied recursively to each subset. Similarly, inducers for decision lists employ a "separate-and-conquer" procedure where one rule is built for a subset of the instances and further rules are generated recursively for the remaining data. Rule generation is guided by a criterion similar to the split selection measure for decision trees. Tests are added to a rule by optimizing this criterion. The aim is to find rules that cover large, pure subsets of instances in order to maximize the empirical support for each rule.

Like the basic divide-and-conquer algorithm, the standard separate-and-conquer procedure cannot deal successfully with noisy data because it aims at identifying pure subsets of instances. To make rule learning useful in real-world domains, some kind of pruning mechanism is essential. There are two main pruning strategies for separate-and-conquer rule learners. The first builds a full unpruned rule set and then simplifies the classifier by eliminating tests from rules or by deleting individual rules. This is done by globally optimizing the rule set according to a pruning criterion. Global pruning of rule sets is related to post-pruning methods for decision trees because a full, unpruned classifier is generated before pruning begins. However, there are some important differences and they will be discussed in Chapter 5.

The second strategy adopts a simpler approach where each rule is simplified immediately after it has been generated. This simplification strategy is called "incremental" pruning, and it turns out that it has several conceptual advantages over the global optimization approach. Considering the rule set as a whole, incremental pruning is similar to pre-pruning in decision trees because rules are pruned before the structure of the full, unpruned rule set is known. However, at the rule level, incremental pruning more closely resembles post-pruning because rules are pruned after they have been fully expanded. The differences between global and incremental pruning will be discussed in more detail in Chapter 5.

1.3 Thesis Structure

This thesis proposes improvements to standard pruning algorithms for decision trees and lists that make extensive use of statistical significance tests. The main claim is:

Significance tests are useful tools for pruning decision trees and lists.

Although some existing pruning methods apply significance tests—mostly in an *ad hoc* fashion—there has been little prior work attempting to establish (a) whether they can be used to improve standard pruning algorithms in a way that is useful in practice, (b) whether there are factors that are important for their successful application, and (c) whether there is any advantage in using non-parametric tests instead of parametric ones.

The argument is structured as follows. Chapter 2 analyzes why pruning is a requirement in virtually all practical data mining applications and shows which components of a classifier are good candidates for pruning because they are likely to be unreliable. A simple quantitative relationship is derived between the expected error rate of a component and its empirical support in the training data. Then we explain why learning algorithms often induce classifiers that contain spurious components and why significance tests suggest themselves as a possible remedy.

Chapter 3 applies significance testing to a standard post-pruning algorithm for decision trees and compares the performance of "parametric" and "non-parametric" tests in this setting. Non-parametric tests have the theoretical advantage that they do not make any potentially restrictive assumptions about the data available for testing. It is shown how the test's significance level can be determined automatically to maximize the benefit obtained from pruning.

Chapter 4 investigates the use of significance tests in pre-pruning algorithms. We evaluate empirically whether it is advantageous to use a non-parametric test instead of a parametric one, and determine whether it is beneficial to locally adjust for significance tests on multiple attributes when pruning. Multiple tests require the significance level to be adjusted appropriately. The question is whether this should be done locally at each node of the tree or whether a global adjustment is sufficient. Post-pruning and pre-pruning are compared on a level footing by employing the same basic statistical technique in conjunction with both pruning paradigms. It has often been claimed that post-pruning is superior to pre-pruning because it has access to information disclosed by building a full decision tree before pruning takes place. However, it appears that this claim has never been evaluated in a fair empirical comparison.

Chapter 5 discusses established pruning methods for rule inducers and identifies their shortcomings. As a consequence of this analysis, we propose a simple incremental pruning algorithm for decision lists that uses the separate-and-conquer scheme to generate a series of pruned decision trees, obtaining one rule from each of them. It is then shown how significance-based rule selection can be used to reduce the size of the induced decision lists, and how significance testing can be integrated into the pruning procedure. We also demonstrate how the induction process can be accelerated by building partial decision trees instead of full ones. The performance of the new pruning procedure is compared to standard incremental pruning, and the effect of significance testing as well as partial tree generation is evaluated empirically. Finally, we determine how well the new rule inducer performs compared to other state-of-the-art rule learners.

Related work is discussed in a separate section at the end of each chapter if it has not been mentioned previously. The main findings are summarized in Chapter 6 and evaluated in light of the thesis statement at the beginning of this section. Chapter 6 also identifies the most important contributions to research and points out potential directions for future work.

1.4 Experimental Methodology

All methods investigated in this thesis are evaluated empirically on benchmark problems from the UCI repository of machine learning datasets (Blake et al., 1998). The same 27 datasets are used throughout. They comprise all the commonly-used practical datasets in the repository, excluding ones with more than 1000 instances due to the sheer number of experiments involved in generating the results presented in this thesis. These datasets and their properties are listed in Table 1.1. They represent a wide range of practical problems. About half of the datasets have binary class labels and the other half represent multi-class domains. Most problems contain a

Table 1.1: Datasets used for the experiments

Dataset	Instances	Missing values (%)	Numeric attributes	Nominal attributes	Classes
anneal	898	0.0	6	32	5
audiology	226	2.0	0	69	24
australian	690	0.6	6	9	2
autos	205	1.1	15	10	6
balance-scale	625	0.0	4	0	3
breast-cancer	286	0.3	0	9	2
breast-w	699	0.3	9	0	2
german	1000	0.0	7	13	2
glass (G2)	163	0.0	9	0	2
glass	214	0.0	9	0	6
heart-c	303	0.2	6	7	2
heart-h	294	20.4	6	7	2
heart-statlog	270	0.0	13	0	2
hepatitis	155	5.6	6	13	2
horse-colic	368	23.8	7	15	2
ionosphere	351	0.0	34	0	2
iris	150	0.0	4	0	3
labor	57	3.9	8	8	2
lymphography	148	0.0	3	15	4
pima-indians	768	0.0	8	0	2
primary-tumor	339	3.9	0	17	21
sonar	208	0.0	60	0	2
soybean	683	9.8	0	35	19
vehicle	846	0.0	18	0	4
vote	435	5.6	0	16	2
vowel	990	0.0	10	3	11
zoo	101	0.0	1	15	7

mix of nominal and numeric attributes, some are purely numeric, and a few are purely nominal. A significant fraction also contain missing values.

As mentioned above, accuracy on the training data is not a good indicator of a classifier's future performance. Instead, an independent sample of test instances must be used to obtain an unbiased estimate. One way of evaluating a learning algorithm is to split the original dataset randomly into two portions and use one portion for training and the other for testing. However, the resulting estimate depends on the exact split that is used and can vary significantly for different splits, especially if the original dataset is small. A more reliable procedure is to repeat the process several times with different random number seeds and average the results.

"Cross-validation" is a slightly more sophisticated version of this basic method for performance evaluation. In a k-fold cross-validation, the training data is split into k approximately equal parts. The first of these k subsets is used for testing and the remainder for training. Then the second subset is used for testing and all other $k-1$ subsets are used for training. This is repeated for all k subsets and the results are averaged to obtain the final estimate. Compared to the naive procedure, cross-validation has the advantage that each instance is used for testing exactly once.

Usually the parameter k is set to ten. It has been found empirically that this choice produces the most reliable estimates of the classifier's true performance on average (Kohavi, 1995b), and there is also a theoretical result that supports this finding (Kearns, 1996). The variance of the estimate can be further reduced by taking the average of a repeated number of cross-validation runs, each time randomizing the original dataset with a different random number seed before it is split into k parts (Kohavi, 1995b). Ideally the cross-validation is performed for all possible permutations of the original dataset. However, this kind of "complete cross-validation" is computationally infeasible for all but very small datasets (Kohavi, 1995a), and must be approximated by a limited number of cross-validation runs. All performance estimates presented in this thesis are derived by repeating ten-fold cross-validation ten times and averaging the results.

When comparing two learning algorithms, the difference in performance is important. The same ten cross-validation runs, using the same ten randomizations of the dataset, can be used to obtain estimates for both schemes being compared. However, to make maximum use of the data we would like to use estimates from a complete cross-validation for the comparison. Fortunately the ten given estimates for the two schemes can be used to get some information on the outcome that would be obtained if they were compared using complete cross-validation, because the mean of a limited number of cross-validation estimates is approximately normally distributed around the "true" mean—the result of a complete cross-validation. Consequently a two-tailed paired t-test (Wild & Weber, 1995) on the outcome of the ten cross-validation runs can be used to test whether the result of a complete cross-validation would be likely to show a difference between the two schemes. In this thesis a difference in performance is called "significant" according to a t-test at the 5% significance level applied in this fashion.

Note that this kind of significance testing does not test whether two schemes perform differently across all possible data samples that could potentially be drawn from the domain. It cannot test for this because all cross-validation estimates involved are based on the same set of data (Salzberg, 1997). It is solely a way of inferring the potential outcome of a complete cross-validation. There exists a heuristic procedure based on repeated cross-validation that attempts to test whether two methods perform differently with respect to all possible data samples from a domain (Dietterich, 1998). However, it is based on an *ad hoc* modification of the standard t-test and does not appear to have been widely adopted in the machine learning community.

1.5 Historical Remarks

Most of this thesis was written in the second half of 1999. A short draft version of Chapter 2, describing the simple mathematical relationship between the size of a model's component and its expected error, was written at the end of 1998. Work on the thesis was interrupted when I became involved in writing a book on machine learning techniques for data mining (Witten & Frank, 2000). Progress on the material for Chapter 3 was made in April, May, and June 1999. Chapter 4 is based on work presented at the Fifteenth International Conference on Machine Learning in Madison, Wisconsin (Frank & Witten, 1998b). Chapter 5 is also based on research presented at the same conference (Frank & Witten, 1998a). However, the material has been revised and significantly extended.

During the course of my PhD studies, I had several opportunities to explore areas in the field of machine learning that are not directly related to the material presented in this thesis. I conducted research on using model trees for classification (Frank et al., 1998), on applying machine learning to automatic domain-specific keyphrase extraction (Frank et al., 1999), on adapting the naive Bayes learning technique to regression problems (Frank et al., in press), and on making better use of global discretization methods for numeric attributes (Frank & Witten, 1999). The techniques described in Chapter 5 also led to an efficient algorithm for learning decision lists in a regression setting (Hall et al., 1999).

Chapter 2

Why prune?

Decision trees and lists split the instance space into disjoint pieces and assign a class label to each of these partitions. Their advantage is that they produce an explicit representation of how this is done. The description of the piece belonging to a particular class can be transformed into disjunctive normal form using standard logical operations. In this form each class is described by a proposition whose premise consists of a disjunctive clause defining the sections of the space belonging to the class. The individual components of the clause are called "disjuncts." In decision trees and lists, disjuncts are mutually exclusive, that is, they do not overlap in instance space.

Here is a disjunctive normal form describing a class C, consisting of a set of disjuncts D_i:

$$D_1 \vee D_2 \vee \ldots \vee D_l \Rightarrow C. \tag{2.1}$$

An instance is assigned to class C if it is covered by any of the disjuncts D_i. Each class can be described by an expression of this form. In decision trees and lists, the premises for the different classes are mutually exclusive and cover the instance space completely, that is, the premise of exactly one of the propositions is satisfied by any one instance in the space. The premises' disjuncts consist of conjunctions of attribute-value tests T_{ij}:

$$D_i := T_{i1} \wedge T_{i2} \wedge \ldots \wedge T_{im_i}. \tag{2.2}$$

In most implementations of decision trees and lists, the T_{ij} are simple univariate tests of the form $A < v$ or $A \geq v$, where A is a numeric attribute and v is a constant, or $A = v$, where A is a nominal attribute and v one of its possible values.

Each disjunct belongs to a class and all future test instances covered by the disjunct are assigned to this class. To minimize the number of incorrect class assignments, the disjunct should be labeled with the class that is most likely to occur. According to the maximum likelihood principle (Mitchell, 1997), which is commonly employed in learning algorithms for decision trees and lists, this is the class that occurs most frequently in the training data. Thus the disjunct is labeled with the majority class of the instances that it covers.

The disjunct's size is the number of training instances pertaining to it.[1] The error rate of a disjunct is the fraction of future test instances that it misclassifies. It seems clear that small disjuncts are more error prone than large ones, simply because they enjoy less support from the training data. A number of authors have observed this fact empirically (Holte et al., 1989; Ting, 1994; Ali & Pazzani, 1995; Weiss, 1995; Van den Bosch et al., 1997; Weiss & Hirsh, 1998). In the following we explain why this is the case and why pruning algorithms are a remedy for disjuncts that are overly error prone.

Section 2.1 demonstrates quantitatively the conditions under which sampling variance causes the expected error of a disjunct to decrease with its size when no search is involved in generating it, and determines situations where these conditions occur in practical classifiers. Section 2.2 explains why search aggravates the problem of sampling variance, causing learning algorithms to generate disjuncts that do not reflect structure in the underlying domain. Section 2.3 identifies pruning mechanisms as a way to eliminate these superfluous components of a classifier. Section 2.4 discusses related work, and Section 2.5 summarizes the findings of this chapter.

2.1 Sampling variance

When a finite sample of instances is drawn from an infinite population, the proportion of instances belonging to each class usually differs between the sample and the population. Only in the limit, when the sample becomes very large, do the observed proportions converge to the true population proportions. With smaller samples, probability estimates derived from the sample proportions can deviate substantially

[1] Note that this definition of a disjunct's size is different from the definition used by Holte et al. (1989), who define the size of a disjunct to be the number of training instances that it correctly classifies.

from the truth. In classification, where performance is measured by the number of correct predictions on future data, this is not a problem so long as the majority class—the one with greatest probability—is the same in the sample as in the population. In that case, assigning the most populous class in the sample to future data will maximize the number of correct predictions. However, accuracy degrades if the majority class in the sample differs from the most likely class in the population. Because of sampling variance this is particularly likely if the sample is very small—as it is in small disjuncts. In the following we investigate the quantitative relationship between the size of a disjunct and its expected classification error when the disjunct has not been fitted to the training data by a search mechanism. For simplicity, we consider the two-class case. All statements can be generalized to the multi-class case in a straightforward way.

Let A and B be the two possible classes and p the "true" proportion of class A instances covered by a particular disjunct, that is, the proportion of class A instances if an infinite amount of data were drawn from the population. Then, if n instances in the training data are covered by the disjunct, the number of class A instances among those n follows a binomial distribution. The binomial distribution models the distribution of the number of heads in n tosses of a biased coin. The probability of observing n_A instances of class A in the sample is

$$\Pr(N_A = n_A) = \binom{n}{n_A} p^{n_A}(1-p)^{n-n_A}. \tag{2.3}$$

The maximum likelihood rule labels the disjunct with the majority class of the training instances that it covers. Given Equation 2.3, the probability that class A is the majority class for a particular set of training instances is

$$\Pr(N_A > \frac{n}{2}) = \sum_{n_A=\lceil \frac{n+1}{2} \rceil}^{n} \Pr(N_A = n_A). \tag{2.4}$$

Correspondingly, the probability that class B is the majority class is

$$\Pr(N_A < \frac{n}{2}) = \sum_{n_A=0}^{\lfloor \frac{n-1}{2} \rfloor} \Pr(N_A = n_A). \tag{2.5}$$

We assume that if both classes are equally represented, the disjunct is labeled by

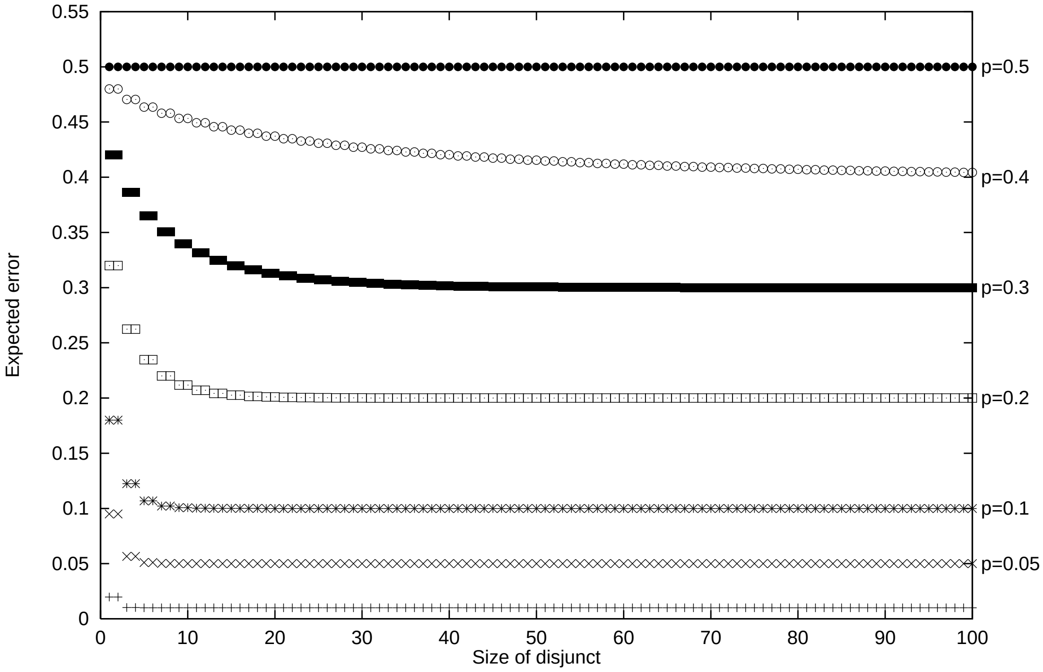


Figure 2.1: Expected error of disjuncts for different sizes and different values of p

flipping an unbiased coin.

We have all the terms needed to compute the disjunct's probability of misclassifying a fresh test instance drawn from the population as a function of its size n and the true population proportion p of class A. This probability—the expected error rate of the disjunct—is

$$\begin{aligned} e(n,p) &= (1-p)\left[\Pr(N_A > \frac{n}{2}) + \frac{1}{2}\Pr(N_A = \frac{n}{2})\right] + && (2.6)\\ & \quad p\left[\Pr(N_A < \frac{n}{2}) + \frac{1}{2}\Pr(N_A = \frac{n}{2})\right] && (2.7)\\ &= (1-p)\Pr(N_A > \frac{n}{2}) + p\Pr(N_A < \frac{n}{2}) + \frac{\Pr(N_A = \frac{n}{2})}{2} && (2.8) \end{aligned}$$

Figure 2.1 depicts a plot of $e(n,p)$ for various values of n and p. Note that $e(n,p) = e(n,1-p)$. The figure shows that for a given p, the expected error decreases as the size of the disjunct increases—unless the data is free of potential ambiguities and p is either zero or one, or neither of the classes is prevalent in the population and p is exactly 0.5. This is a direct consequence of the fact that the smaller the disjunct, the more likely it is that the majority class observed from the training data is not the "true" majority class in the population.

There are two factors, potentially occurring simultaneously, that can cause the true population proportion p to be different from zero or one:

1. The disjunct covers an area of the instance space that is "noisy";
2. The disjunct covers an area of the instance space that belongs to more than one class.

Both factors have the same effect because the disjunct's expected error depends only on the value of p that they cause and the number of instances covered by the disjunct.

Webster's dictionary (Merriam-Webster, 1999) defines noise to be: "... irrelevant or meaningless data or output occurring along with desired information." In this context, desired information can be interpreted as an instance exhibiting the class label that is most likely to occur at the point in instance space where it is located—in other words, an instance with the "correct" class assigned to it. An instance with an "incorrect" class is viewed as noise. The presence of noise means that the disjunct cannot possibly be divided into disjoint regions that describe the target concept with 100% accuracy because there is a non-zero probability that the same instance will exhibit different class labels if it is sampled multiple times from the domain. There are two possible reasons for this that can potentially coincide:

- uncertainty in the class value of an instance ("class noise"), and
- uncertainty in the attribute values of an instance ("attribute noise").

Uncertainty in the class or attribute values is either an inherent feature of the domain or is introduced in the process of measuring a particular value.

If the disjunct covers regions of the instance space that exhibit different majority classes in the population, it could potentially be split into smaller pieces so that each piece corresponds to one of these distinct regions. However, the classification model fails to separate the different regions. This also means that p is different from zero or one.

2.2 Overfitting

The above analysis assumes that the proportion of class A instances in a disjunct follows a binomial distribution. This is indeed the case if the classifier has not been fitted to the training data—in other words, if it has been generated without looking at the training instances' class labels. However, algorithms for classification learning use the class labels to obtain a classifier. They seek a model that accurately explains the training data. To this end they search the space of possible classifiers for a candidate with low error rate on the given observations. In other words, they attempt to identify regions of the instance space that contain "pure" clusters of training instances—clusters that contain instances of only one class—and construct disjuncts describing them. This means that the proportion of class A training instances in a disjunct is no longer binomially distributed: it is strongly biased towards being either zero or one.

The more search is involved in generating a classifier, the more likely it becomes that the observed class proportions are optimistically biased. Search increases the likelihood of finding apparent patterns that are caused by sampling variance and do not reflect the true underlying population proportions. This phenomenon is called "oversearching" (Quinlan & Cameron-Jones, 1995). It is a direct consequence of the fact that learning algorithms perform multiple comparisons of potential classifiers, resulting in the most promising candidate being chosen at each stage of the search process (Jensen & Cohen, 2000). The more candidates considered, the greater the chance of detecting a classifier that performs particularly well just by chance.

The following example illustrates the problem. Consider a domain where the true proportion p of class A instances is 0.25 at every point in instance space. The space consists of a single numeric attribute. In this case the expected error on fresh data would be minimized if class B were predicted for every test instance. Now consider the expected error rate of a randomly generated classifier with ten disjuncts for different numbers of training instances. The classifier is generated by randomly splitting the instance space into ten pieces so that each piece contains at least one training instance. The curve that results from averaging the expected error of 100 random classifiers with 100 randomly sampled training sets is the lower one of the two curves depicted in Figure 2.2. The errorbars are 95% confidence intervals on the mean. The figure shows that the random classifier's performance falls short of the optimum error rate of 0.25 if the amount of training data is small. Approximately 200 instances are required to achieve close to optimum performance.

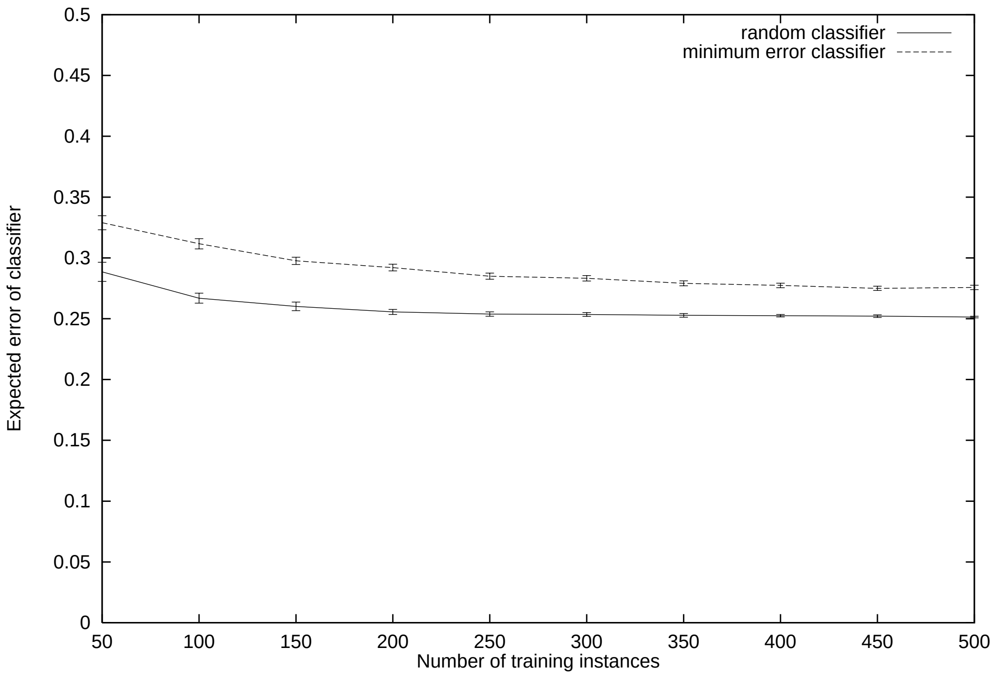


Figure 2.2: Average error of random classifier and minimum error classifier

Because the classifier is built by randomly partitioning the instance space, this loss in performance is due solely to the effect of sampling variance discussed in the previous section.

Now consider the average performance of a minimum error classifier with 10 disjuncts using the same 100 random training sets. A minimum error classifier exhibits the lowest possible error rate on the training data for the given number of disjuncts. For a split on a single numeric attribute it can be found efficiently using a dynamic programming technique that effectively searches through all possible multiway splits on the given attribute (Fulton et al., 1995). The resulting curve, also depicted in Figure 2.2, shows that this classifier performs even worse than the random model from above. Note that both classifiers have the same "complexity" because they both contain the same number of disjuncts. Hence the degradation in performance can only be a result of the search process required to find the minimum error classifier.

Figure 2.3 shows the average expected error for disjuncts of different sizes for both the randomly generated classifier and the minimum error classifier when 100 training instances are used. To obtain accurate statistics these results were generated from 10000 randomly sampled training sets. The expected error rates of the disjuncts in

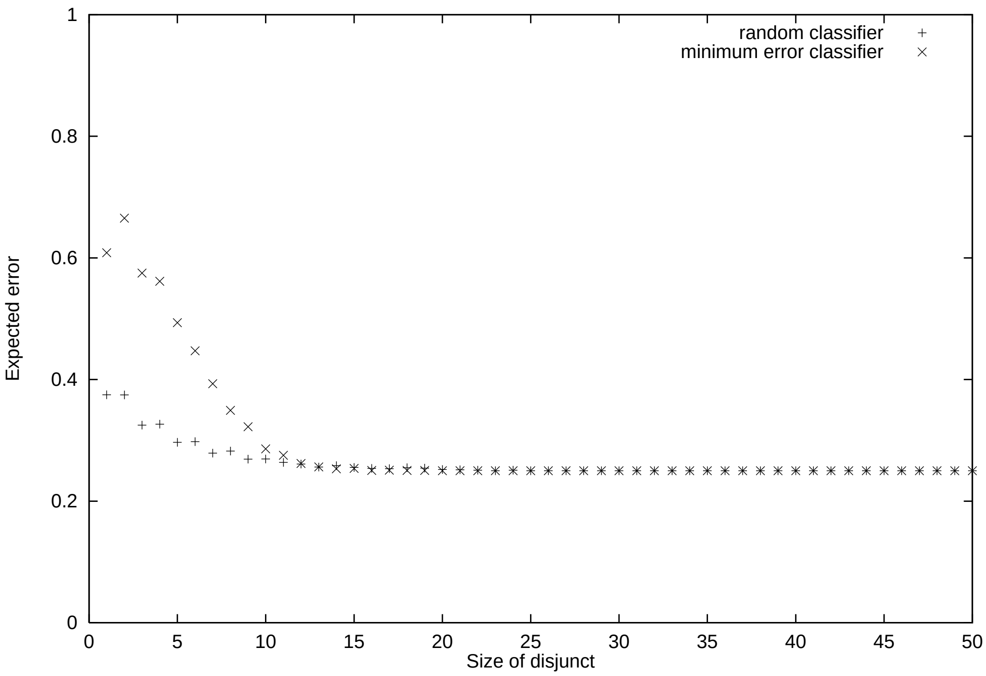


Figure 2.3: Average error of different size disjuncts for random classifier and minimum error classifier

the random classifier closely approximate the values predicted by Equation 2.6 from the last section. (The values predicted by Equation 2.6 are not shown in Figure 2.3 because they are virtually identical to those produced by the random classifier.) However, the small disjuncts in the minimum error classifier perform substantially worse than those from the random one. Only for large disjuncts the discrepancy vanishes. This demonstrates that search aggravates the problem of small disjuncts because it increases the likelihood that chance patterns with little support in the data are awarded a separate disjunct in the classifier.

Figure 2.4 depicts how frequently on average the two algorithms generate disjuncts of a particular size. It shows that search skews the distribution of disjunct sizes: the minimum error classifier is more likely to generate very small disjuncts (less than five instances) and less likely to produce slightly larger ones (between 5 and 25 instances) than the random classifier. Combined with the fact that search causes the smallest disjuncts to be particularly error prone, this finding explains why the expected performance of the minimum error classifier is worse than that of the random classifier.

Both classifiers fit the training data overly closely. This phenomenon is called "overfitting." As the above results demonstrate, overfitting depends on two factors:

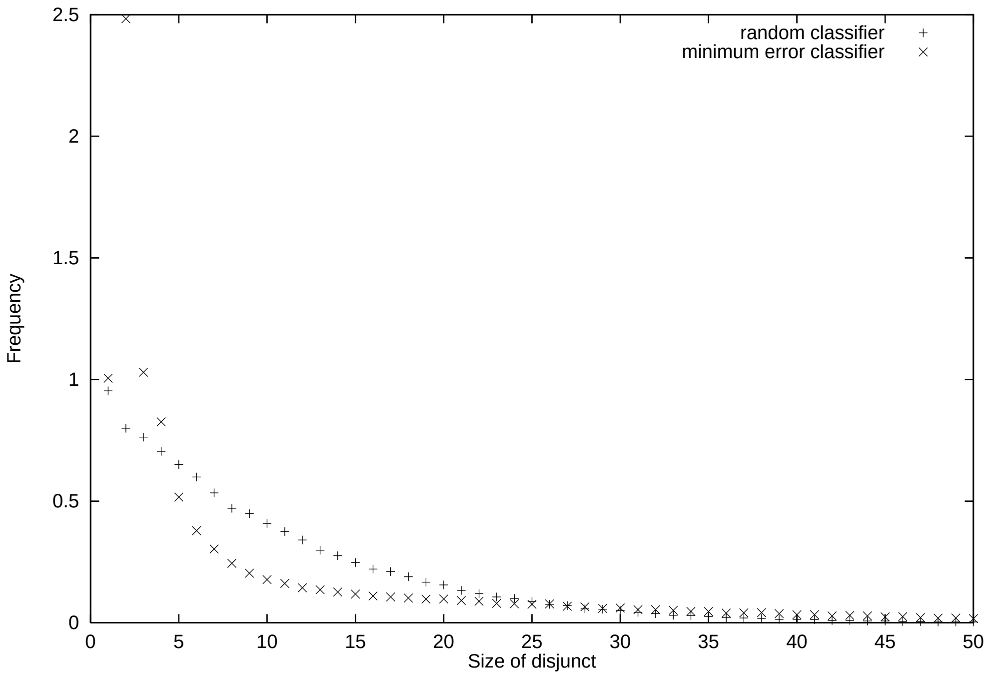


Figure 2.4: Average number of different size disjuncts for random classifier and minimum error classifier

the "complexity" of the classifier and the amount of search involved in generating it from the training data. In decision trees and lists, complexity and search are often linked because induction algorithms for these classifiers proceed according to a "general-to-specific" paradigm: they start with the simplest possible classifier and iteratively extend it by adding more and more disjuncts until the training data is fully explained.

Although the potential for complex classifiers combined with intensive search can lead to overfitting, it is often a requirement for finding an accurate classifier. Thus, rather than restricting the learning algorithm's search space *a priori*, it is advisable to eliminate unreliable components of a classifier before the model is put to use. The process of removing parts of a model is called "pruning," and efficient pruning methods combat overfitting by removing those parts that are due to random fluctuations in the training data.

2.3 Pruning

Pruning simplifies a classifier by merging disjuncts that are adjacent in instance space. This can improve the classifier's performance by eliminating error-prone components. It also facilitates further analysis of the model for the purpose of knowledge acquisition. Of course, pruning should never eliminate predictive parts of a classifier. Therefore the pruning procedure requires a mechanism for deciding whether a set of disjuncts is predictive or whether it should be merged into a single, larger disjunct.

This thesis investigates pruning algorithms that employ statistical significance tests to make this decision. In terms of a significance test, the pruned disjunct represents the "null hypothesis" and the unpruned disjuncts represent the "alternative hypothesis." The test is a means of determining whether the data provides sufficient evidence in favor of the alternative. If this is the case, the unpruned disjuncts remain in place; otherwise, pruning proceeds.

Significance tests have an intuitive explanation: they test whether the observed association between a set of disjuncts and the data is likely to be due to chance alone. They do so by computing the probability of obtaining a chance association that is at least as strong as the observed association if the null hypothesis were correct. If this probability does not exceed a given threshold—in other words, if the observed association is unlikely to be due to chance—then the unpruned disjuncts are assumed to be truly predictive; otherwise the model is simplified. The aggressiveness of the pruning procedure is controlled by the threshold employed—the test's "significance level."

If a significance level close to zero is used for pruning, only parts of a classifier that are very likely to be predictive will survive the pruning process. In many cases this is the appropriate strategy to obtain a classifier that is accurate on future data. However, when the aim is to maximize accuracy, there are situations where it is better to accept less significant associations. Consider the case where the true population proportions for all disjuncts in an unpruned classifier are zero or one. Any kind of pruning will decrease the classifier's accuracy on future data. Assume that the individual disjuncts have very little support from the training data. If a low significance level is used in this situation, many disjuncts will be pruned because they have too little support to be statistically significant.

This example shows that the significance level must be adjusted to the properties of the domain to obtain optimum performance on future data. This issue will be discussed in more detail in Section 3.2.1. There is another factor that requires the significance level to be modified: the number of significance tests that are performed during pruning. The more tests that are performed, the more likely it becomes that a significant association is observed by chance alone. Consequently the significance level must be adjusted in a situation with multiple significance tests (Westfall & Young, 1993; Jensen & Schmill, 1997). Chapter 3 presents a technique that can be used to automatically find the significance level that maximizes the classifier's predictive accuracy on future data.

Significance tests are a means of combating the problem of sampling variance in induction. However, they assume that the data samples involved are unbiased and have not been used in fitting the model. Hence the training data can usually not be used to perform the test. The only exception is when the data has not been involved in building the model—for example if the model has been selected randomly, or if the fitting process can be taken into account when the test is performed. Normally, a fresh sample of data is required. A standard procedure in pruning methods is to hold out some of the training data when the model is built so that unbiased estimates for the pruning process can be obtained from the hold-out data. The same procedure can be used in conjunction with significance tests and is a basic ingredient of most of the pruning algorithms investigated in this thesis.

2.4 Related Work

There is a theoretical result stating that all learning algorithms perform equally well when averaged across all possible domains. For every domain where learning algorithm A performs better than algorithm B, there is another domain where it performs worse. This result is known as the "no free lunch theorem" (Wolpert, 1996), or the "conservation law for generalization performance" (Schaffer, 1994). It implies that pruned and unpruned classifiers perform equally well averaged across all possible domains.

The no free lunch theorem is based on two assumptions. The first assumption is that the test data for every domain only contains instances that are not represented in the training set—in other words, that performance is measured according to the off-training set (OTS) error (Wolpert, 1996). This differs from the standard setting that is being considered in this thesis, where the test data is sampled according to the same probability distribution as the training data and the same instance can occur in both sets. In this standard setting the training and test datasets are independent identically distributed (IID)—in other words, performance is measured according to the IID error—and there are learning schemes that perform better than others on average. For example, a classifier that predicts the majority class in the training data will generally outperform a classifier that predicts a random class label. However, the difference between OTS and IID error is very small in most practical applications because the probability that a randomly chosen test instance occurs in the training set is very close to zero (Wolpert, 1996).

The second assumption is more restrictive. It presupposes that all domains are equally likely to occur. This is obviously not correct in the real world: practical domains usually exhibit a certain degree of smoothness and the underlying relationships are quite simple (Holte, 1993). This means that the "overfitting avoidance bias" implemented by pruning strategies often improves performance in practical applications (Schaffer, 1993). However, there are domains where pruning methods are likely to decrease the accuracy of a classifier, even in the IID setting (Schaffer, 1993). For example, according to the discussion from Section 2.1, pruning decreases a classifier's accuracy if the true population proportions are zero or one for each of its disjuncts.

Schaffer (1991) investigates the conditions under which a simple classifier outperforms a more complex one by looking at the simplest possible case: a situation with a binary class where the complex classifier consists of two disjuncts corresponding to the values of a single binary attribute and the simple classifier is a single disjunct that comprises the whole instance space. No search is involved in generating the classifiers and both values of the binary attribute are equally likely. Schaffer restricts his attention to training sets of size three and only those situations where the complex classifier has lower error on the training data than the simple one, assuming that otherwise the simple classifier is preferable. Let the true population proportions of the first of the two possible classes in the two disjuncts be p_1 and p_2. Assuming that p_1 and p_2 are uniformly distributed, he shows that the complex classifier is more accurate than the simple one on average, and that for the majority of combinations of p_1 and p_2 values it is better to choose the complex classifier on average across all training sets. However, he also shows that additional attribute noise can substantially increase the fraction of combinations where the simple model outperforms the complex one, and that this happens to a lesser extent with class noise as well. Note that Schaffer's analysis does not involve a pruning algorithm that actively chooses between the simple and the complex model according to the training data. It just compares the expected performance of the two classifiers and identifies combinations of p_1 and p_2 values where pruning is appropriate.

Kim and Koehler (1994) extend Schaffer's results using the same two models by considering constraints on the distribution of class values in the training data and larger training sets. They find that the simple model outperforms the complex one for a larger fraction of (p_1, p_2)-tuples when the skewness of the distribution of class values or the amount of training data increases. They also show that higher levels of additional class noise than those considered by Schaffer cause the simple model to outperform the complex one more frequently.

In another paper, Schaffer (1992) shows empirically that the effect of pruning depends on the amount of training data relative to the complexity of the domain, in other words, the "sparseness" of the data. He concludes that it is often better not to prune a learned classifier at all if training data is sparse. He also shows that a skewed class distribution can increase the benefit obtained from pruning. These results are consistent with Kim and Koehler's theoretical findings.

The expected error rate of a learning algorithm is a random variable that depends on the particular data sample being used for training. A statistical tool called the "bias plus variance decomposition" divides this random variable into three parts: "bias," "variance," and "noise" (Kohavi & Wolpert, 1996). As above, noise is the error component that is due to the uncertainty inherent in the domain. It cannot be overcome by any learning algorithm. Bias measures the component that does not depend on the particular data sample at hand. Variance denotes the component that is due to the particular training set being used. If a learning algorithm's predictions do not depend on the particular training set, the variance is zero. In terms of the bias plus variance decomposition, pruning methods are designed to decrease the expected error by reducing its variance component. Ideally they do so without increasing the bias. There are several ways to define the bias plus variance decomposition for classification learning in exact mathematical terms (Dietterich & Kong, 1995; Kohavi & Wolpert, 1996; Breiman, 1996; Tibshirani, 1996; Friedman, 1997). However, it has yet to be determined which of these definitions is the most appropriate.

2.5 Conclusions

This chapter explains why pruning methods are an important ingredient of practical learning algorithms for decision trees and lists. We have shown quantitatively how sampling variance affects the expected error of a disjunct when no search is involved in generating it, and have identified conditions—for example noise in the domain—which cause sampling variance to degrade a classifier's accuracy. This result explains why small disjuncts are inherently more error prone than large ones. Then we discussed why search aggravates the problem of error-prone disjuncts, and how the complexity of a classifier and the search involved in generating it can cause overfitting—resulting in chance disjuncts that are due solely to sampling variance and do not represent structure in the underlying domain. Finally we pointed out why pruning strategies—in particular, pruning strategies based on significance tests—are an appropriate way of dealing with the problem of superfluous disjuncts.

Chapter 3

Post-pruning

The aim of pruning is to discard parts of a classification model that describe random variation in the training sample rather than true features of the underlying domain. This makes the model more comprehensible to the user, and potentially more accurate on new data that has not been used for training the classifier. When pruning, an efficient mechanism is needed for distinguishing parts of a classifier that are due to chance effects from parts that describe relevant structure.

Statistical significance tests are theoretically well-founded methods for determining whether an observed effect is a genuine feature of a domain or just due to random fluctuations in the sampling process. Thus they can be used to make pruning decisions in classification models. Reduced-error pruning (Quinlan, 1987a), a standard algorithm for post-pruning decision trees, does not take statistical significance into account, but it is known to be one of the fastest pruning algorithms, producing trees that are both accurate and small (Esposito et al., 1997). This chapter investigates whether significance tests can be used to improve on this well-known pruning procedure. As we will see, the main problem is choosing an appropriate significance level for each individual pruning task.

The primary hypothesis is the following:

Hypothesis 3.1. *Reduced-error pruning generates smaller and more accurate decision trees if pruning decisions are made using significance tests and the significance level is chosen appropriately for each dataset.*

Significance tests can be divided into those called "parametric tests" that make some mathematical assumptions about the underlying distribution function, and those called "non-parametric tests" (Good, 1994) that are essentially assumption-free. Tests based on the chi-squared distribution, for example, belong to the former group: they assume that the test statistic is distributed according to the chi-squared distribution. Their use is questionable for small sample sizes because then the assumptions required for applying the chi-squared distribution cease to be valid. Permutation tests, on the other hand, make no assumptions about the functional form of the underlying distribution, and belong to the second group of tests. Consequently, they can be applied with any sample size.

Graceful behaviour for small samples is particularly important in learning algorithms like decision tree inducers, where pruning decisions must be made for smaller and smaller subsets of data. Given these considerations, it is plausible that, for a given amount of pruning, decision trees pruned using a permutation test will be more accurate than those pruned using a parametric test. With both types of tests, the amount of pruning, and therefore the size of the resulting trees, can be fine-tuned by adapting the significance level. This leads to our secondary hypothesis:

Hypothesis 3.2. *If decision tree A is the result of pruning using a permutation test, and decision tree B is the result of pruning using a parametric test, and both trees have the same size, then A will be more accurate than B on average.*

The structure of this chapter is as follows. Section 3.1 explains why it is important to consider statistical significance when pruning decisions are made. Section 3.2 details standard statistical tests that can be employed. These tests are compared on artificial and practical datasets in Section 3.3 and 3.4 respectively. Section 3.5 shows how the significance level can be chosen to minimize tree size without loss in accuracy, and Section 3.6 discusses related work. Section 3.7 summarizes the findings of this chapter.

3.1 Decision tree pruning and statistical significance

Figure 3.1 depicts an unpruned decision tree. We assume that a class label has been attached to each node of the tree—for example, by taking the majority class of the training instances reaching that particular node. In Figure 3.1 there are two classes: A and B.

The tree depicted in Figure 3.1 can be used to predict the class of a test instance by filtering it to the leaf node corresponding to the instance's attribute values and

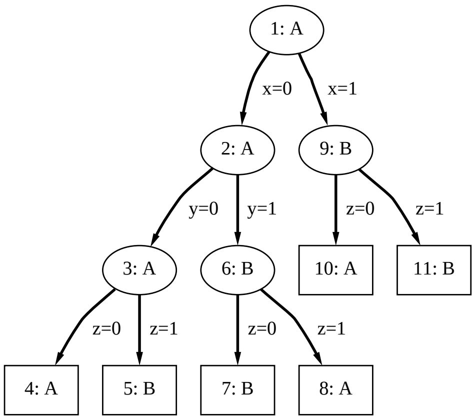


Figure 3.1: A decision tree with two classes A and B (with node numbers and class labels)

x	y	z	class
0	0	1	A
0	1	1	B
1	1	0	B
1	0	0	B
1	1	1	A

Figure 3.2: An example pruning set

assigning the class label attached to that leaf. However, using an unpruned decision tree for classification potentially "overfits" the training data: some of its structure, namely that due to random variation in the particular data sample used for training, might not be warranted. Consequently, it is advisable, before the tree is put to use, to ascertain which parts truly reflect effects present in the domain, and discard those that do not. This process is called "pruning."

A general, fast, and easily applicable pruning method is "reduced-error pruning" (Quinlan, 1987a). The idea is to hold out some of the available instances—the "pruning set"—when the tree is built, and prune the tree until the classification error on these independent instances starts to increase. Because the instances in the pruning set are not used for building the decision tree, they provide a less biased estimate of its error rate on future instances than the training data. Reduced-error pruning uses this estimate as a guideline for its operation.

Figure 3.2 shows an example pruning set for the decision tree from Figure 3.1. Figure 3.3 displays how reduced-error pruning proceeds for this example. In each tree, the number of instances in the pruning data that are misclassified by the

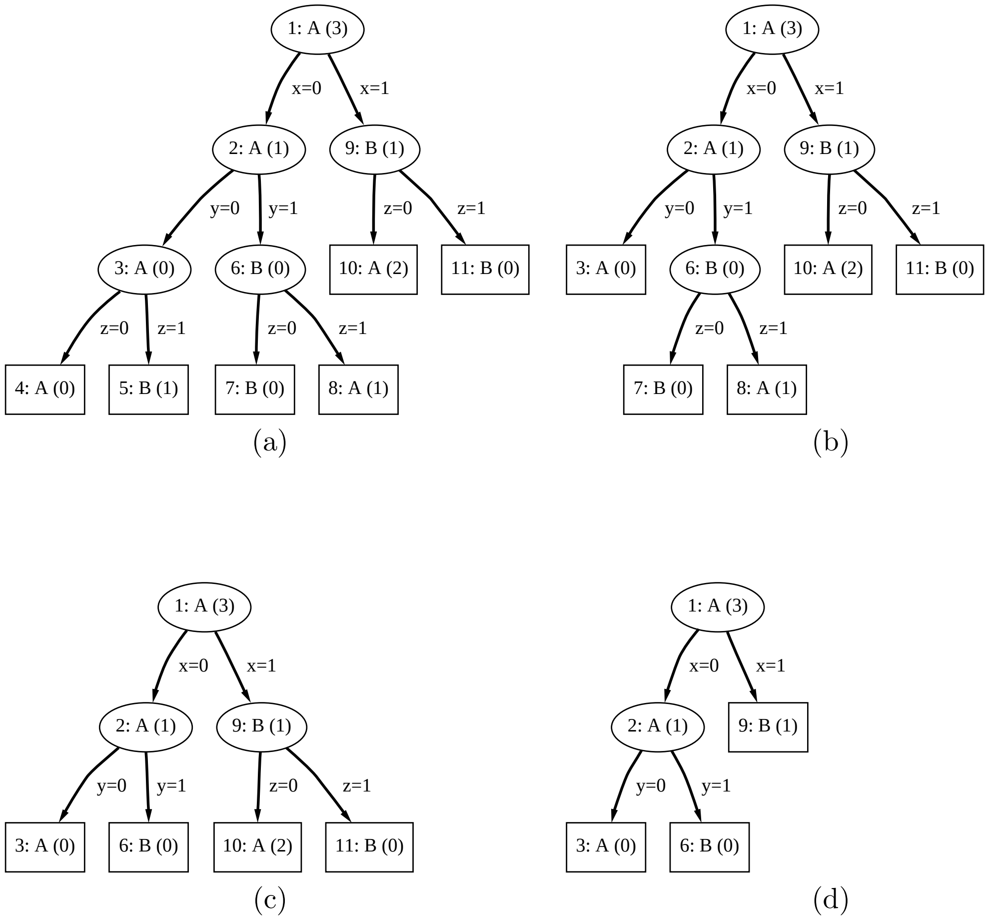


Figure 3.3: Reduced-error pruning example

individual nodes are given in parentheses. A pruning operation involves replacing a subtree by a leaf. Reduced-error pruning will perform this operation if it does not increase the total number of classification errors. Traversing the tree in a bottom-up fashion ensures that the result is the smallest pruned tree that has minimum error on the pruning data (Esposito et al., 1995). This traversal strategy is a direct result of the condition that a node can only be converted to a leaf if all subtrees attached to it have already been considered for pruning.

Assuming that the tree is traversed left-to-right, the pruning procedure first considers for removal the subtree attached to node 3 (Figure 3.3a). Because the subtree's error on the pruning data (1 error) exceeds the error of node 3 itself (0 errors), node 3 is converted to a leaf (Figure 3.3b). Next, node 6 is replaced by a leaf for the same reason (Figure 3.3c). Having processed both of its successors, the pruning procedure then considers node 2 for deletion. However, because the subtree attached to node 2 makes fewer mistakes (0 errors) than node 2 itself (1 error), the subtree remains in place. Next, the subtree extending from node 9 is considered for pruning, resulting in a leaf (Figure 3.3d). In the last step, node 1 is considered for pruning, leaving the tree unchanged.

Unfortunately, there is a problem with this simple and elegant pruning procedure: it overfits the pruning data. This phenomenon was discovered by Oates and Jensen (1997). The consequence is the same as for overfitting the training data, namely an overly complex decision tree. A simple example shows why overfitting occurs.

Consider a dataset with 10 random binary attributes with uniformly distributed values 0 and 1. Suppose the class is also binary, with an equal number of instances of each class, and class labels A and B. Of course, the expected error rate for this domain is the same for every possible classifier, namely 50%, and the simplest conceivable decision tree for this problem—predicting the majority class from the training data—consists of a single leaf node. We would like to find this trivial tree because we could then correctly deduce that none of the attributes in this dataset provides any information about the class label associated with a particular instance.

We applied reduced-error pruning to this problem, using a randomly generated sample of 100 instances. The information gain criterion (Quinlan, 1986b) was employed for selecting the tests at each of the tree's nodes. Two thirds of the data were used to grow the initial unpruned tree and the remaining third was set aside for pruning—the standard procedure for pruning a classifier using a hold-out set (Cohen, 1995; Fürnkranz, 1997; Oates & Jensen, 1999). Figure 3.4 shows the unpruned decision tree. The number of instances in the pruning data that are misclassified by the tree's individual nodes are given in parentheses. Figure 3.5 shows the same tree after pruning has taken place.

Figure 3.5 suggests that, although reduced-error pruning successfully reduces the size of the unpruned tree, it certainly does not generate the minimal decision tree. This hypothesis can easily be confirmed by repeating the experiment with different randomly generated datasets (Jensen & Schmill, 1997). Figure 3.6 summarizes the results obtained by repeating it 100 times for each of 10 different training set sizes. The error bars are 95% confidence intervals for the mean of the decision trees' size; they show that reduced-error pruning indeed generates overly complex decision trees

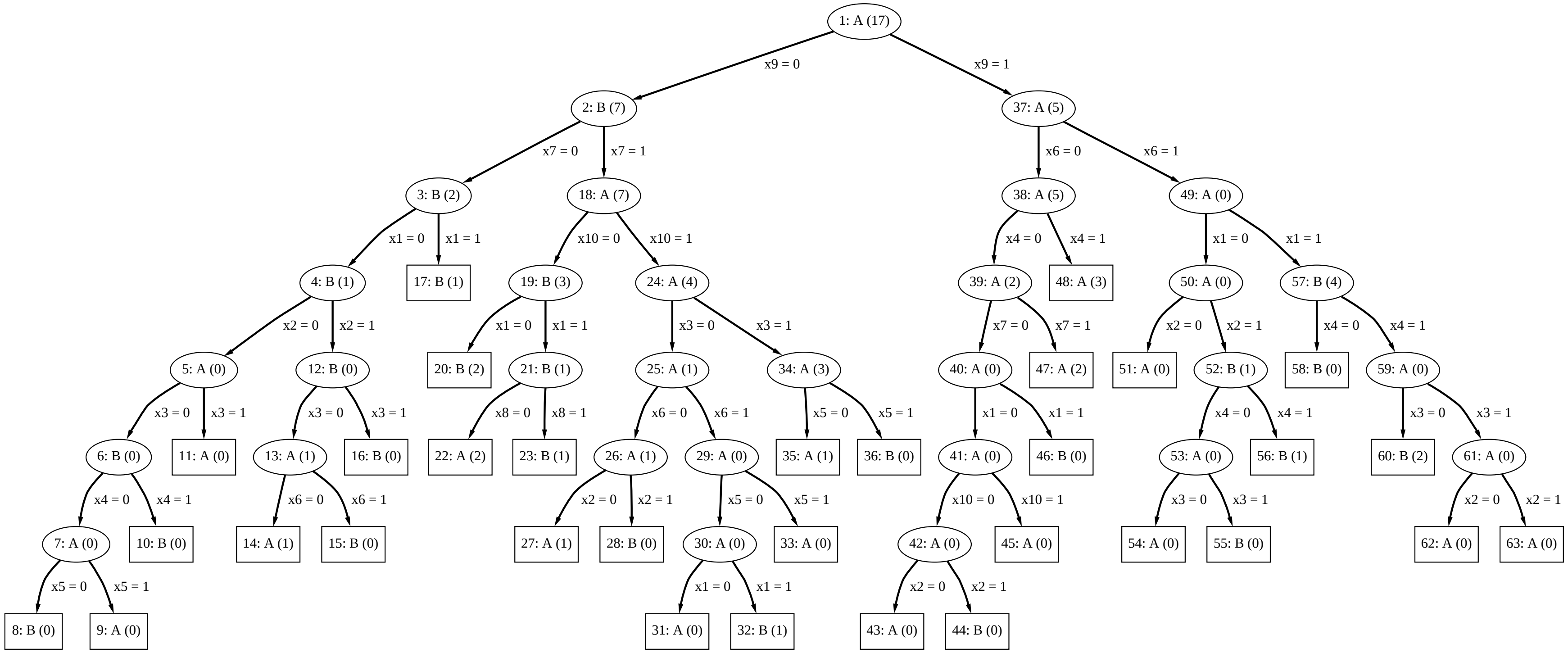


Figure 3.4: Tree for no-information dataset before pruning

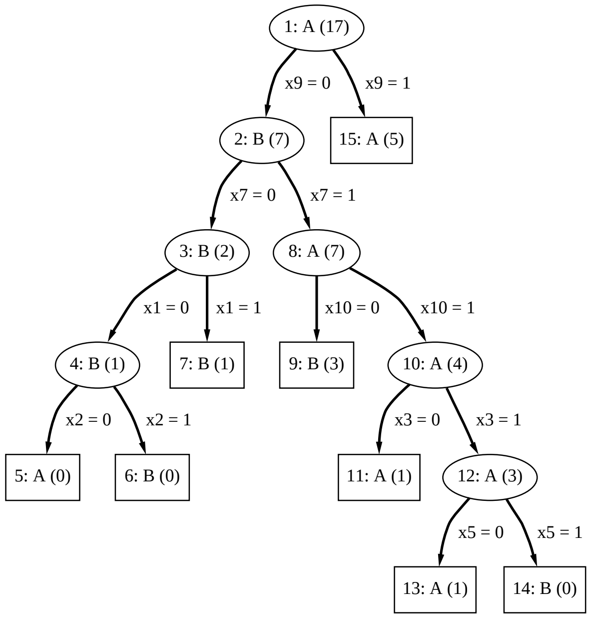


Figure 3.5: Tree for no-information dataset after pruning

for this problem—independent of the particular training set.

A closer look at Figure 3.4 reveals the reason for this pathology. Because of the large number of subtrees that must be considered for pruning, there are always some that fit the pruning data well *just by chance*. The pruning procedure incorrectly retains those trees. This also explains why the size of the pruned tree increases with the number of training instances: the larger the unpruned tree, the more subtrees are likely to fit the pruning data well by chance. The problem arises because reduced-error pruning does not take into account the fact that sampling variance can cause the observed majority class at a particular node to be incorrect even if the data is not used during training. As discussed in Sections 2.1 and 2.2 of Chapter 2, the distribution of class values at the nodes of a decision tree does not necessarily reflect the true distribution, and this effect is particularly pronounced if the data samples at the nodes are small. Stated differently, the pruning procedure does not test whether the association between the predictions and the observed class values in the pruning data is statistically significant or due solely to sampling variance.

Statistical significance tests—more specifically, significance tests on contingency tables—are an obvious remedy: a subtree is only retained if there is a significant association between its predictions and the class labels in the pruning data. The following sections discuss these tests in detail, and compare their performance in

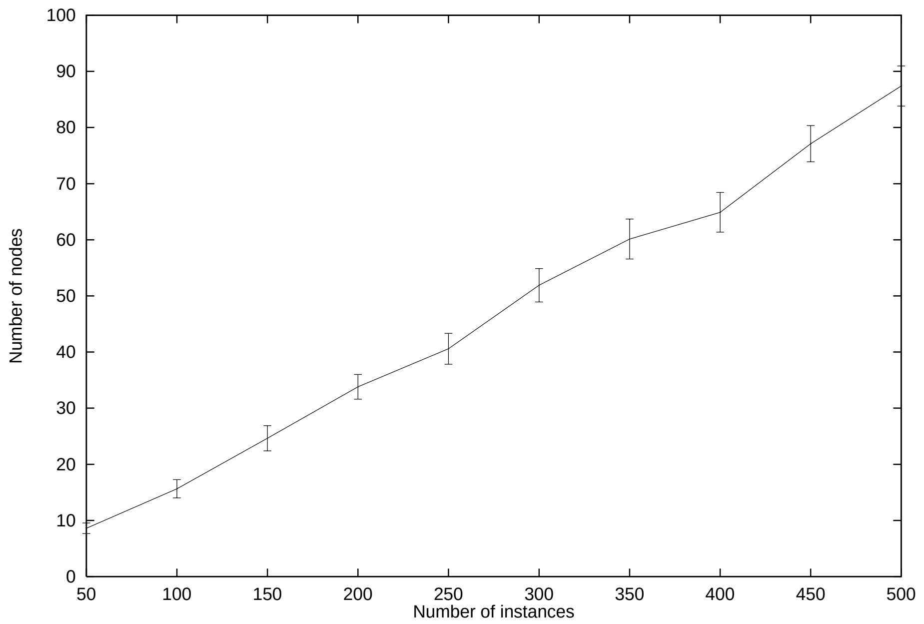


Figure 3.6: Size of pruned tree relative to training set size, using reduced-error pruning

pruning decision trees derived from artificial and practical datasets.

3.2 Significance tests on contingency tables

Tests for independence in a contingency table determine whether there is a statistically significant dependence between the values of two nominal variables. In the pruning problem above, the two variables are (a) the actual class values in the pruning data, and (b) the class values predicted by the subtree. We want to know whether there really is a significant dependence between the true class values and the predicted ones, or whether it is likely that the observed correlation is just due to chance—that is, caused by random fluctuations in the particular sample of pruning data being employed.

Figure 3.7 shows the structure of a contingency table. The I rows and J columns correspond to the values of the two nominal variables being considered. Each cell of the table contains the number n_{ij} of times the corresponding combination of values has been observed in N instances. The row and column totals N_{i+} and N_{+j} are the sums of the entries in each row and column respectively.

In the evaluation of classification algorithms, a contingency table comparing the

n_{11}	n_{12}	$\dots$	n_{1J}	N_{1+}
n_{21}	n_{22}	$\dots$	n_{2J}	N_{2+}
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
n_{I1}	n_{I2}	$\dots$	n_{IJ}	N_{I+}
N_{+1}	N_{+2}	$\dots$	N_{+J}	N

Figure 3.7: Structure of a contingency table

x	y	z	actual class	predicted class
0	0	1	A	A
0	1	1	B	B
1	1	0	B	B
1	0	0	B	B
1	1	1	A	B

Figure 3.8: Example pruning set with predicted class values

algorithm's predictions to the actual class values is known as a "confusion matrix." Figure 3.9 shows the confusion matrix for the final pruned tree from Figure 3.3d. It summarizes the actual and predicted class values for the pruning data that are listed in Figure 3.8. The column totals represent the number of pruning instances of each class that reach the corresponding node—in this case, node 1. The row totals correspond to the number of pruning instances that would be assigned to each class if the corresponding subtree—in this case, the full tree—were used for classification. A confusion matrix makes it easy to see how many pruning instances would be correctly classified by a subtree: the number of correctly classified instances is the sum of the diagonal elements in the matrix. Confusion matrices, as a particular type of contingency table, are the basis for the significance tests considered in this chapter.

		actual class values: A	B	
predicted class values	A	1	0	1
	B	1	3	4
		2	3	5

Figure 3.9: Example confusion matrix

3.2.1 Testing significance

The hypothesis that two variables, such as the actual and predicted class values, are independent is called the "null hypothesis," and a significance test determines whether there is enough evidence to reject this hypothesis. When pruning decision trees, rejecting the null hypothesis corresponds to retaining a subtree instead of pruning it. The degree of association between the two variables is measured by the "test statistic." The significance test computes the probability that the same or a more extreme value of the statistic will occur by chance if the null hypothesis is correct. This quantity is called the "p-value" of the test. If the p-value is low, the null hypothesis can be rejected, that is, the observed degree of association is unlikely to be due to chance. Usually, this is done by comparing the p-value to a fixed significance level α, rejecting the null hypothesis if α is at least as large as the p-value. A significance test can be applied to the pruning problem by computing the p-value for the observed association and comparing it to α, retaining or discarding the subtree accordingly.

Two quantities are important when evaluating a significance test: the probabilities of committing a "Type I" and a "Type II" error. A Type I error occurs when the null hypothesis is incorrectly rejected, in other words, when a significant association is found although a real association does not exist. The probability of committing a Type I error is the significance level α, and usually fixed *a priori*. In the context of pruning, committing a Type I error corresponds to incorrectly deciding not to prune—also called "underpruning." A Type II error occurs when the null hypothesis is incorrectly accepted, in other words, when a significant association is overlooked. The probability of a Type II error is denoted by β. Committing a Type II error corresponds to "overpruning"—deciding to discard a subtree although it would be better if it were kept. The complement of the probability of a Type II error, $1 - \beta$, is called the "power" of the significance test. For a fixed significance level α, the power depends on several factors (Cliff, 1987).

- *The sample size:* The power of a test increases with the amount of data available because more data increases the likelihood of detecting an association.

- *The strength of the effect being tested:* A strong association is more likely to be spotted than a weak one.

- *The type of test employed:* Not all tests are equally powerful— some incur larger β values than others.

The power $1 - \beta$ is also a function of the significance level α (Cliff, 1987). The smaller the significance level, the less powerful the test. There is a trade-off between the two types of error: the less likely a Type I error becomes, the more likely a Type II one, and vice versa. Ideally, α would be zero, and $1 - \beta$ would be one. In that case, both types of error would be eliminated. Unfortunately this is not feasible if the test is based on a finite amount of data.

Significance tests are often used to test the result of a scientific experiment, in other words, to determine whether there is a significant association between a proposed theory and experimental data. In this application, it is particularly important that the Type I error is small, because this type of error directly affects how often an incorrect theory is found to be correct. This is why a small significance level, often 0.05, is commonly required when experimental results are published in scientific journals. On the other hand, no constraints are placed on the Type II error. It is up to the researcher to maximize the test's power for the required significance level. This can, for example, be achieved by collecting a sufficient amount of data.

The task of testing scientific hypotheses differs fundamentally from the problem of pruning classification models. When significance tests are used for pruning, the aim is to maximize accuracy on future data, and both types of error are equally important. The problem is to find the right balance between α and β in order to avoid underpruning as well as overpruning. The correct balance hinges on the three factors listed above. This means that the optimum significance level depends on, among other things, the amount of data available for the learning problem at hand. Using the same, fixed significance level for every domain is not the right thing to do.

This issue is independent of the well-known fact that α must also be adjusted for multiple significance tests (Jensen & Schmill, 1997). Adjustments for multiple tests are necessary because pruning usually requires more than one test to be performed and the likelihood of detecting a chance association increases with the number of tests involved. This problem will be discussed in more detail in Chapter 4.

It appears that the necessity of balancing α and β has been consistently overlooked in previous approaches that apply significance tests in learning algorithms. Unfortunately there is little hope of finding an analytic solution to this problem because it depends on the strength of the underlying effect, which is usually unknown. Section 3.5 presents a way of selecting a good value for α by cross-validation.

Note that, throughout this thesis, we assume that the same value for α is appropriate for every region of the instance space. It is plausible that further improvements are possible by adjusting the significance level locally to each region—for example, because different regions usually contain different amounts of data. However, it is much harder to choose an appropriate α-value independently for each region. Moreover, this choice is necessarily based on less data and therefore likely to be less reliable. Thus we restrict our attention to the global approach.

Statistical tests are based on the distribution of the test statistic under the null hypothesis. As mentioned above, they can be divided into two groups: parametric tests, which rely on the assumption that the distribution belongs to a particular class of parametric functions, and non-parametric tests, which do not require the distribution function to have any particular form. The next subsection discusses parametric tests based on the chi-squared distribution, and following that we present a group of non-parametric tests known as "permutation tests."

3.2.2 Parametric tests

The most popular tests for independence in contingency tables are based on the fact that some test statistics have approximately a chi-squared distribution with $(I-1)(J-1)$ degrees of freedom if the null hypothesis is correct. The classic test statistic with this property is the chi-squared statistic (Agresti, 1990)

$$\chi^2 = \sum_i \sum_j \frac{(n_{ij} - e_{ij})^2}{e_{ij}}, \tag{3.1}$$

where e_{ij} are the expected cell counts under the null hypothesis, calculated according to

$$e_{ij} = N\hat{p}_i\hat{p}_j = N\frac{N_{i+}}{N}\frac{N_{+j}}{N} = \frac{N_{i+}N_{+j}}{N}, \tag{3.2}$$

where $\hat{p}_i$ is the estimated probability that a particular observation will fall into row i, and $\hat{p}_j$ is the corresponding probability for column j. Because these two probabilities are independent under the null hypothesis, their product constitutes the probability that an observation will fall into cell (i, j).

The expected cell counts for the example table in Figure 3.9 are $e_{11} = 2/5$, $e_{12} = 3/5$, $e_{21} = 8/5$, and $e_{22} = 12/5$. Thus the χ^2-value for this table is

$$\frac{(1-0.4)^2}{0.4} + \frac{(0-0.6)^2}{0.6} + \frac{(1-1.6)^2}{1.6} + \frac{(3-2.4)^2}{2.4} = 1.875. \tag{3.3}$$

An alternative to the chi-squared statistic, which also has a chi-squared distribution, is the "log likelihood ratio" (Agresti, 1990)

$$G^2 = 2 \sum_i \sum_j n_{ij} \log (n_{ij}/e_{ij}). \tag{3.4}$$

A disadvantage of tests based on the chi-squared distribution is that they are statistically invalid when the sample size is small (Agresti, 1990). The chi-squared distribution is an approximation to the test statistics' true sampling distribution under the null hypothesis, and this approximation is only accurate when the sample is large. Unfortunately, there is no single rule that can be used to determine when the approximation is valid (Agresti, 1990, page 247). Cochran (1954), for example, suggests that a test based on the χ^2 statistic can be employed if none of the expected cell counts is smaller than 1, and at most 20% of them have expected values below 20. Agresti (1990, page 247) writes that "...the chi-squared approximation tends to be poor for sparse tables containing both small and moderately large expected frequencies." However, it has also been shown empirically that χ^2 works better with small sample sizes and sparse tables than G^2 (Agresti, 1990, page 246). Depending on the expected cell counts, using the chi-squared distribution in conjunction with G^2 can result in a test that is either too conservative or too liberal (Agresti, 1990, page 247). A test that is too conservative produces p-values that are too large, while one that is too liberal produces p-values that are too small.

Predicted	True
A	A
B	B
B	B
A	A

Table 3.1: Predicted and true class labels for a small dataset

	Permutations																							
	A_1	A_1	A_1	A_1	A_1	A_1	B_2	B_2	B_2	B_2	B_2	B_2	B_3	B_3	B_3	B_3	B_3	B_3	A_4	A_4	A_4	A_4	A_4	A_4
	B_2	B_2	B_3	B_3	A_4	A_4	A_1	A_1	B_3	B_3	A_4	A_4	A_1	A_1	B_2	B_2	A_4	A_4	A_1	A_1	B_2	B_2	B_3	B_3
	B_3	A_4	B_2	A_4	B_3	B_2	B_3	A_4	A_1	A_4	A_1	B_3	B_2	A_4	A_1	A_4	A_1	B_2	B_3	B_2	A_1	B_3	A_1	B_2
	A_4	B_3	A_4	B_2	B_2	B_3	A_4	B_3	A_4	A_1	B_3	A_1	A_4	B_2	A_4	A_1	B_2	A_1	B_2	B_3	B_3	A_1	B_2	A_1
χ^2	4	0	4	0	0	0	0	4	0	0	4	0	0	4	0	0	4	0	0	0	0	4	0	4

Table 3.2: Permutations of the dataset from Table 3.1 and corresponding χ^2-values

3.2.3 Non-parametric tests

Non-parametric tests have the advantage that they do not make assumptions about the specific functional form of the test statistic's distribution. Permutation tests are a class of non-parametric tests that compute the test statistic's distribution under the null hypothesis explicitly, by enumerating all possible permutations of the given data. The best-known member of this group of tests is Fisher's exact test for 2×2 contingency tables (Agresti, 1990). Unlike parametric tests using the chi-squared distribution, permutation tests are statistically valid in small-sample situations (Good, 1994). They are based on the fact that, under the null hypothesis, all possible permutations of a dataset are equally likely to occur. The p-value of a permutation test is the fraction of these permutations for which the test statistic has an equally or more extreme value than for the original data (Good, 1994).

In the case of classification problems, permutations of a dataset correspond to permutations of the class labels associated with its instances (Jensen, 1992). Table 3.1 shows the predicted and true class labels for an example dataset with four instances, and Table 3.2 lists all 24 possible permutations of the true class labels. Note that there are permutations that produce identical columns in Table 3.2. Consider, for example, the permutation that just swaps the class labels of the two instances belonging to class A (the 22nd permutation in Table 3.2): it produces the same column as the original data (the first permutation in Table 3.2).

Each permutation can be written as a contingency table by pairing it with the

		actual		
		A	B	
pre-	A	2	0	2
dicted	B	0	2	2
		2	2	4

		actual		
		A	B	
pre-	A	1	1	2
dicted	B	1	1	2
		2	2	4

		actual		
		A	B	
pre-	A	0	2	2
dicted	B	2	0	2
		2	2	4

Figure 3.10: Contingency tables for permutations in Table 3.2

predicted class values from Table 3.1. Some of the permutations map to the same table. The 24 permutations from Table 3.2 result in three tables, depicted in Figure 3.10. According to Equation 3.1, the χ^2-value for the first table is four, for the second one zero, and for the third one also four. Table 3.2 lists these χ^2-values for the individual permutations. Like the original data, 8 of the 24 permutations result in a χ^2-value of four. Hence, if a permutation test were performed on the dataset from Table 3.1 using χ^2 as the test statistic, the p-value would be 8/24, or 1/3.

The three tables in Figure 3.10 have one thing in common: they all share the same marginal totals. Permuting the class labels does not alter the number of instances that belong to each class; neither does it alter the number of instances assigned to each class by the classifier. In statistical terms, permutation tests on contingency tables derive a p-value by conditioning on the given marginal totals.

Identical contingency tables result in the same value for the test statistic. Thus the p-value of a permutation test can also be computed by summing up the probabilities of all contingency tables with an equally or more extreme value for the test statistic. The probability of a contingency table p_f—equivalent to the fraction of random permutations resulting in the table—can be written in closed form

$$p_f = \frac{\prod_i n_{i+}! \prod_j n_{+j}!}{n! \prod_i \prod_j n_{ij}!}. \tag{3.5}$$

This function is known as the multiple hypergeometric distribution (Agresti, 1990). If s_f is the test statistic's value for the contingency table f, and s_o its value for the original data, the p-value can be written as

$$p = \sum I(s_f \leq s_o) p_f, \tag{3.6}$$

where $I()$ is the indicator function, and the sum is over all contingency tables with the same marginal totals.

Unfortunately, both methods of computing the exact p-value of a permutation test—enumerating all possible permutations directly, or just enumerating all possible contingency tables—are computationally infeasible for all but very small sample sizes. For some test statistics sophisticated "network algorithms" have been developed that only evaluate a small subset of all possible contingency tables in order to compute the exact p-value (Good, 1994). They make use of mathematical properties of the test statistic in order to cut down the search space. However, even these sophisticated algorithms are still computationally very expensive and only applicable if the sample size is small.

3.2.4 Approximation

The solution to this dilemma is to realize that the "exact" p-value is not required when performing the test. It is sufficient to approximate it to a degree that makes it clear beyond reasonable doubt whether it is greater than the significance level α or not. This can be done by randomly sampling permutations from the space of all possible permutations, and computing the proportion $\hat{p}$ of these for which the test statistic has an equally or more extreme value than for the original data (Good, 1994). The proportion $\hat{p}$ constitutes an approximation to the exact p-value, and the precision of this approximation increases with the number of random samples that are generated. Alternatively, one could sample from the space of all possible contingency tables and use Equation 3.6 to approximate the exact p-value. This is advantageous if extra speed is important, because there are efficient algorithms for generating random contingency tables with a given set of marginal totals (Patefield, 1981).

It remains to determine how many random samples are needed to give an approximation that is accurate enough. Statisticians have designed a procedure for this purpose, called the "sequential probability ratio test" (Lock, 1991). Figure 3.11 summarizes the decision rules for this test, where n is the number of random samples. It employs three constants that determine how closely the approximation emulates the exact test: p_a, p_o, and A, where $0 < p_a < \alpha < p_o < 1$, and $0 < A < 1$. Recommended values are $p_a = 0.8 \times \alpha$, $p_o = 1.2 \times \alpha$, and $A = 0.1$ (Lock, 1991).

If $\hat{p} \times n \geq c \times n + \dfrac{\log A}{\log K}$	$\rightarrow$	accept null hypothesis
If $\hat{p} \times n \leq c \times n - \dfrac{\log A}{\log K}$	$\rightarrow$	reject null hypothesis
Otherwise	$\rightarrow$	continue sampling
$c = \log\left(\dfrac{1-p_o}{1-p_a}\right) / \log K$		$K = \dfrac{p_a(1-p_o)}{p_o(1-p_a)}$

Figure 3.11: Sequential probability ratio test

In principle, any test statistic can be employed in conjunction with a permutation test, and this is one of its major advantages. When testing for independence in a contingency table, the test statistic should measure the degree of association between two nominal variables. Depending on the particular type of association being investigated, different test statistics sometimes lead to different results.

3.2.5 Test statistics

Two possible statistics have already been discussed in the context of parametric tests: χ^2 and G^2. Both can also be used to form permutation tests (Good, 1994). The significance level is simply the fraction of random permutations for which the statistic's value is at least as large as for the original data—because both statistics increase monotonically with the degree of association that is present. The chi-squared distribution, which is the basis for the parametric tests discussed earlier, is in fact only an approximation to the permutation distribution of the two statistics, and, as mentioned above, this approximation is unreliable for small sample sizes (Agresti, 1990). Note that, although the sequential probability ratio test is only an approximation to the exact test, it is guaranteed to closely approximate the true p-value, whereas this is not the case for tests based on the chi-squared distribution.

Another potential test statistic has also been mentioned above, although it did not play the role of a test statistic. The probability p_f of a contingency table under the null hypothesis—given by the multiple hypergeometric distribution—is an alternative to χ^2 or G^2 (Good, 1994). The idea is that a rare contingency table, having a low value for p_f, indicates a strong association between the two variables involved. The test's significance level is the fraction of random permutations for

2	0	2
0	2	2
2	2	4

p-value=1/3

1	1	2
1	1	2
2	2	4

p-value=1

0	2	2
2	0	2
2	2	4

p-value=1/3

Figure 3.12: Three tables and their p-values

which p_f is no greater than for the original data—because the greater the association, the smaller the probability. When both variables in the contingency table are binary, this permutation test is known as the two-sided version of Fisher's exact test (Agresti, 1990). In the general case it is sometimes called the Freeman and Halton test (Good, 1994).

All permutation tests share the disadvantage that the distribution of p-values is very sparse when the sample size is extremely small (Agresti, 1990).[1] This is due to the small number of contingency tables that are possible when there are very few instances.

3.2.6 Sparseness

The problem of sparseness is illustrated in Figure 3.12. It shows the three possible contingency tables with two instances in each row and column, and the p-values of Fisher's exact test for each one: 1/3, 1, and 1/3 respectively. It can be shown that the large gaps between the individual p-values have the consequence that the actual probability of committing a Type I error can be much lower than the significance level α (Agresti, 1990). This causes the test to become overly conservative.

There is a solution to this problem known as "randomization on the boundary" (Agresti, 1990). Let p_1 be the p-value of the contingency table under investigation, and p_2 be the next smaller p-value of a table with the same marginal totals (or zero if p_1 is the smallest p-value possible). Moreover, suppose α is equally large or larger than p_2. (Otherwise, the null hypothesis will be accepted.) Randomization on the boundary means that the null hypothesis will be rejected with probability $(\alpha - p_2) / (p_1 - p_2)$ even if α is smaller than p_1—which would normally mean that it will be accepted. In the example from Figure 3.12, randomization on the boundary will reject the null hypothesis for the leftmost table at a significance level of

[1] Note that statisticians use the term "discrete" rather than "sparse."

$\alpha = 1/10$ with probability $3/10$; in other words, it will reject the null hypothesis in 30% of the cases that would normally cause it to be accepted. It can be shown that Fisher's exact test is uniformly most powerful among all unbiased tests for comparing binomial populations, if randomization on the boundary is performed (Agresti, 1990).

An approximation to this randomization procedure is the "mid-p value" method (Lancaster, 1961). Here, the point half-way between p_1 and p_2, $(p_1 + p_2)/2$, is used instead of p_1. The mid-p values for the three tables from Figure 3.12 are 1/6, 2/3, and 1/6 respectively. The mid-p value has the advantage that it is more uniformly distributed under the null hypothesis than the ordinary p-value, and its expected value is 0.5—these are standard properties of significance tests with *continuous* p-values. Statisticians recommend the mid-p procedure to avoid problems arising from sparseness, and argue that it is a good compromise between a conservative test and performing randomization on the boundary (Agresti, 1990)—hence we will use it.

3.3 Experiments on artificial data

The overall qualities of different pruning methods can best be judged by looking at two extremes in the space of potential learning scenarios: on the one hand, a situation where all pruning is beneficial; and on the other, a situation where any kind of pruning is harmful. Focusing on these two cases, this section compares the performance of the tests from the previous section using artificially generated data.

An artificial problem that requires a maximum amount of pruning has already been introduced in Section 3.1. In this problem, the class is completely independent of the predictor attributes, and a single root node is sufficient to achieve optimum predictive performance. As discussed in Section 3.1, reduced-error pruning fails to identify the null model as the correct model for this learning problem. How do the significance tests from the previous section fare?

Figure 3.13 shows how successful they are in identifying and eliminating subtrees that are retained incorrectly by reduced-error pruning. Each time reduced-error pruning decides to retain a subtree, this decision is verified using a significance test.

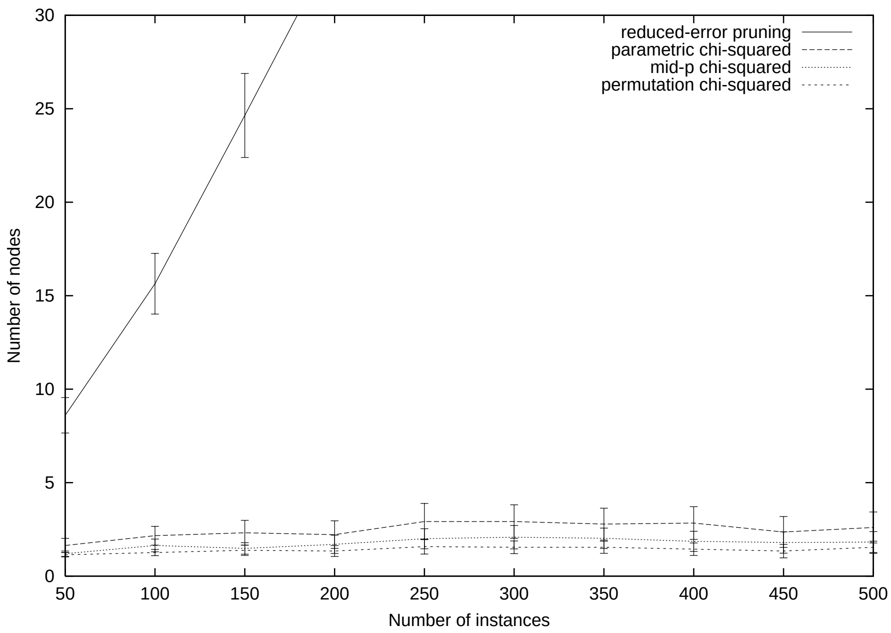


Figure 3.13: Performance of tests for no-information dataset

As in Figure 3.6, 100 different random datasets were generated to obtain one data point in the graph, and the error bars are 95% confidence intervals on the mean. The figure shows graphs for the ordinary parametric chi-squared test and the permutation test based on the chi-squared statistic. For the chi-squared permutation test, it also shows the results obtained using the mid-p adjustment. The graphs for the Freeman and Halton test are omitted because they are almost identical to those for the chi-squared permutation test. For all permutation tests involved, the sequential probability ratio test was used to determine how many permutations to investigate before the null hypothesis could be accepted or rejected. In all tests—the parametric tests as well as the non-parametric ones—a significance level α of 0.1 was used.

The results in Figure 3.13 show that significance testing successfully reduces the number of subtrees that are incorrectly retained by the pruning procedure. Compared to Figure 3.6, the average number of nodes in the pruned decision trees is dramatically reduced. However, there is some variation in the degree to which the different significance tests achieve this reduction. The most liberal is the parametric chi-squared test: it consistently produces trees with the greatest average number of nodes—that is, it incorrectly rejects the null hypothesis more often than the other tests. Considering the tests in order of their performance, the next best is the mid-p

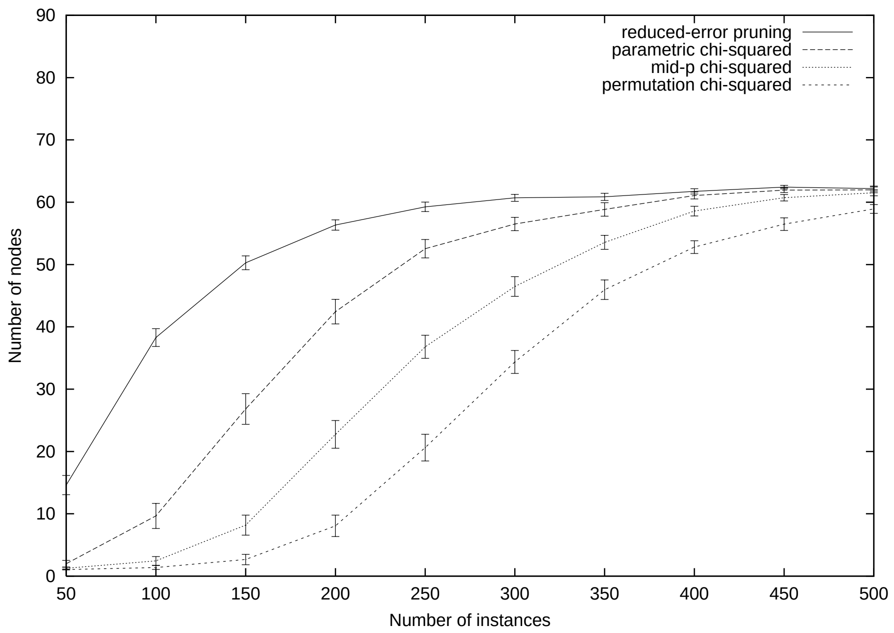


Figure 3.14: Performance of tests for parity dataset

test based on the chi-squared statistic. As expected from the preceding discussion, the mid-p value results in a test that is more liberal than the ordinary permutation test—the best-performing test in this particular task. Considering the definition of the problem, this finding comes as no surprise, because the dataset is devoid of any structure whatsoever; hence no test can be too conservative, and the "best" test is one that always accepts the null hypothesis. Clearly, this is not useful in practice: the question is how sensitive is the test when structure is present.

In order to investigate this question, we move to the other end of the problem space, and consider a dataset where the best way of pruning is not to prune at all: a dataset with a binary class whose value is 1 if and only if the number of ones in the (binary) attributes is even. This is known as the "parity problem," and the correct tree contains exactly one leaf for each possible attribute combination. Again, instances are generated by randomly sampling attribute values from the uniform distribution, followed by assigning the appropriate class value. The pruned tree's expected classification accuracy on fresh data increases monotonically with the number of leaves in the tree. Since the total number of nodes in a (binary) tree (including the leaves) is twice the number of leaves minus one, the expected accuracy also increases monotonically with the number of nodes.

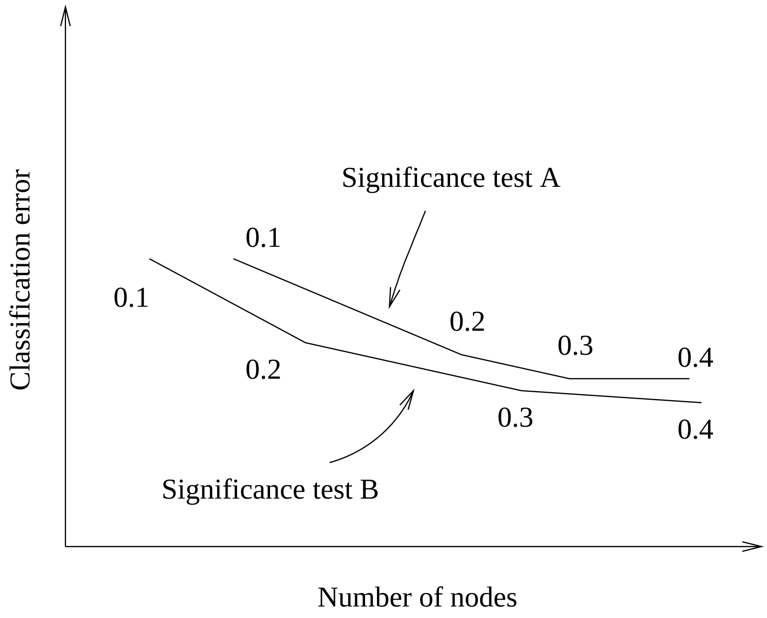


Figure 3.15: Significance test A uniformly dominates test B

Figure 3.14 shows the results for the different pruning strategies, derived in the same way as for Figure 3.13. Again, the Freeman and Halton test is omitted because its results are almost identical to those for the permutation test based on the chi-squared statistic. The performance of the tests is as expected from the previous scenario: again, the parametric chi-squared test is the most liberal, in this case producing the best result, followed by the mid-p permutation test, which gives the second-best result, and the ordinary permutation test based on the chi-squared statistic, which performs worst.

The results for these two artificial scenarios show that, for a given significance level, the different significance tests can be ordered according to the amount of pruning that they incur. There remains the question of whether this difference in behaviour can be eliminated by adjusting the significance level for each test according to the properties of the domain. It is, for example, possible that the parametric chi-squared test at a significance level s_1 results in the same amount of pruning as the chi-squared permutation test at a significance level s_2—if s_1 is set to some appropriate value smaller than s_2.

This hypothesis can be tested by plotting the error rate and tree size for each significance level as depicted in Figure 3.15. In this hypothetical situation, there are two tests A and B, and four significance levels: 0.1, 0.2, 0.3, and 0.4. Assuming that the tests are sufficiently well-behaved, performance at intermediate levels can be interpolated by connecting the data points for these four significance levels with straight lines.

In this contrived example, it is clear that test A really is fundamentally different from test B: for all potential tree sizes, A produces less accurate results than B. In other words, there is no reason to use test A because B always performs better if the significance level is chosen appropriately. However, it is unlikely that the situation is so clear-cut for practical datasets.

3.4 Experiments on practical datasets

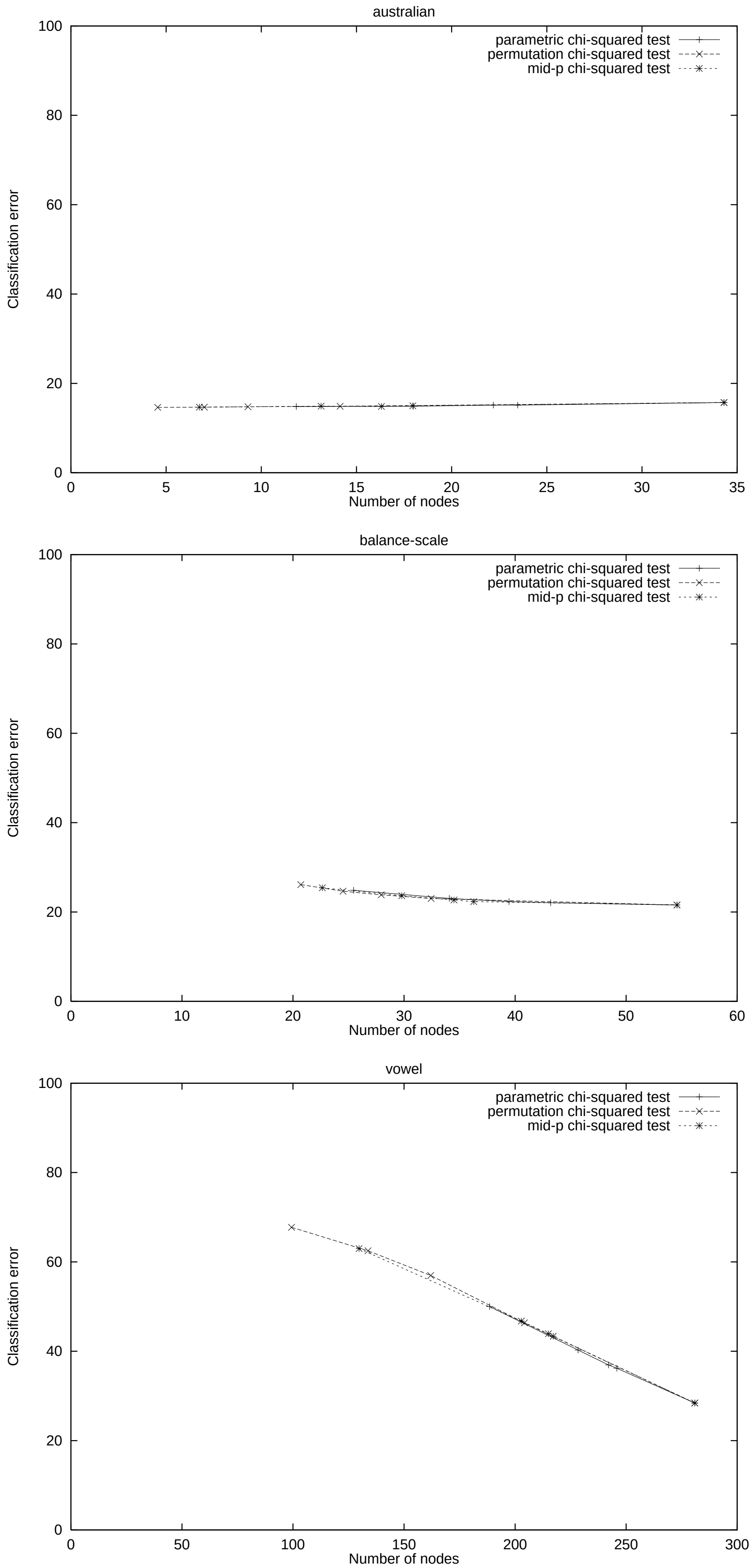
Experiments on datasets from the real world are the only way to evaluate whether a particular method is likely to perform better than other methods in practice. This section presents experimental results for the 27 benchmark datasets from Chapter 1 in order to compare the performance of the statistical tests in a more realistic setting.

For each dataset, a fully grown tree is post-pruned using reduced-error pruning in conjunction with a significance test. Each time reduced-error pruning decides to retain a subtree, this is verified using the significance test and the subtree is retained or discarded accordingly. Two-thirds of the data are used to grow the unpruned tree, and the remaining third is used for pruning. The data is stratified before it is split so that the class distribution is approximately the same in both subsets. For all algorithms, exactly the same random, stratified splits are used to generate the two subsets. The information gain criterion is employed for selecting the tests at each node. Missing attribute values are dealt with in the simplest possible way by assigning them to the most populated branch—both during training and testing. During training this is done before the information gain is computed.

As mentioned above, it is not enough to compare the performance of a set of significance tests at a particular, fixed significance level α, because it is possible that any performance difference vanishes when α is adjusted appropriately for each test. Diagrams like the one in Figure 3.15 are the only reliable way to detect whether a particular test consistently makes "better" pruning decisions than other tests.

Figure 3.16 contains diagrams for three of the 27 datasets (the remaining 24 di-

(a)



(b)

(c)

Figure 3.16: Comparison of significance tests

agrams can be found in Appendix A.1). These examples have been chosen because they represent typical cases. Each diagram contains results for three different significance tests: the parametric chi-squared test, the chi-squared permutation test, and the mid-p version of the latter one. The graphs for the Freeman and Halton test are omitted because they are almost identical to those for the permutation tests based on the chi-squared statistic. As in Figure 3.15, the graphs contain data points for the following four significance levels: 0.1, 0.2, 0.3, and 0.4. We used these four values because we anticipated that even moderate significance levels incur substantial pruning due to the fact that only a fraction of the full dataset is available for pruning. The data points represent estimates from ten-fold cross-validation repeated ten times. As in Figure 3.15, they are connected by straight lines in increasing order. The graphs also contain results for standard reduced-error pruning, which corresponds to applying the significance tests with a significance level of 1. This is the rightmost data point in each of the diagrams, and it is shared by all three of the curves representing the different significant tests. For each test, the data points corresponding to the four significance levels are ordered left to right because they lead to increasingly larger decision trees (as in Figure 3.15).

The graphs present strong evidence that, in practice, the fundamental behaviour of the three tests is almost identical. The only distinguishing feature is the amount of pruning that is being done for a particular significance level. It is safe to conclude that, in the context of post-pruning decision trees, the three tests closely approximate each other if the significance level is chosen appropriately. The figures also show that, as in the artificial examples from the previous section, the tests can be ordered according to how aggressively they prune, given a particular fixed significance level. This follows from the relative ordering of graph symbols (left to right). The permutation chi-squared test is the most aggressive test, followed by its mid-p version, followed by the parametric chi-squared test.

A second result is that significance testing often does not decrease the trees' accuracy despite the fact that it reduces their size. The reduction in tree size is particularly dramatic for the australian (Figure 3.16a), horse-colic (Figure A.1b), breast-cancer (Figure A.1c), pima-indians (Figure A.1c), heart-h (Figure A.1g), hepatitis (Figure A.1g), and vote (Figure A.1g) datasets. In each of these, the estimated number of nodes is reduced by at least 50% if the most aggressive kind of significance testing is performed. For the breast-cancer dataset, the maximum reduction is close to 94% (3.1 instead of 48.6 nodes), indicating that this dataset contains very little information. Even when the most liberal test is used for pruning—in other words, the parametric chi-squared test at a significance level of 0.4—the resulting trees are often significantly smaller.

Sometimes, for example in the australian dataset of Figure 3.16a, significance testing increases accuracy. However, the gain is small. In at least three cases, namely vowel (Figure 3.16c), autos (Figure A.1a), and zoo (Figure A.1e), pruning with significance tests decreases accuracy. However, mild pruning often produces negligible loss of accuracy, and significant loss only occurs with more aggressive pruning. Examples where aggressive pruning is harmful are the balance-scale (Figure 3.16b), audiology (Figure A.1a), glass-2 (Figure A.1b), primary-tumor (Figure A.1c), and soybean (Figure A.1d) datasets.

3.5 Minimizing tree size

The size of a pruned decision tree can be controlled by adjusting the significance level of the statistical test used for pruning. However, the experiments in the previous section show that there is a trade-off: if the significance level is set too small, accuracy declines markedly—and sometimes no additional pruning at all is warranted. Ideally, we want to prune the tree to the point just before accuracy starts to deteriorate, and no further. However, as the results from the previous section show, the "best" significance level depends on the particular properties of the domain—for example, the amount of noise that is present. Hence the appropriate value must be found individually for each domain.

This is an optimization problem that occurs very frequently in machine learning: a parameter setting is required that optimizes predictive performance on future test data. The standard solution is to derive a cross-validation estimate of the accuracy for each parameter setting, and choose the setting that maximizes this estimate (Kohavi, 1995b). Once the optimum value of the parameter has been determined, the learning algorithm is then applied to the full training dataset using this value.

Breiman et al. (1984) use this technique to select an appropriate pruning parameter for pruning decision trees. In this case, there is one additional complication. Often parameter setting s_1 produces a significantly smaller classifier than parameter setting s_2, at the cost of a small but statistically insignificant drop in accuracy (Breiman et al., 1984). Because of statistical fluctuations in the cross-validation estimate, this can occur even if the expected accuracies for the two parameter settings are in fact the same. The experimental results from the previous section exhibit several datasets where this is likely to occur—all those where the graph is parallel to the x-axis. A standard way of circumventing this problem is to consider all parameter values for which the estimated error lies within one standard error of the smallest error estimate observed, and choose the one that produces the smallest trees (Breiman et al., 1984). When pruning with significance tests, the size of the tree decreases with the significance level. Thus optimization chooses the smallest significance level that produces a tree whose error is within one standard deviation of the most accurate tree.

The remainder of this section discusses results obtained by applying this procedure—both with and without the one-standard-error rule—in conjunction with the parametric chi-squared test. As before, the significance test is applied whenever reduced-error pruning decides to retain a subtree. During optimization, seven significance levels are considered: 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, and 1. Using the last value in this list, all subtrees are considered significant, and the effect is the same as for standard reduced-error pruning. The values 0.1, 0.2, 0.3 and 0.4 were used for the experiments in the previous section. The first two values are included because they are frequently used in scientific publications and the experimental results from the previous section show that sometimes even more aggressive pruning is beneficial.

Table 3.3 shows the accuracies for the 27 datasets, and Table 3.5 shows the tree sizes measured in number of nodes. As well as results for standard reduced-error pruning, the tables include results for pruning with significance testing, both without (SIG) and with (SIG-SE) the one-standard-error rule. All estimates are derived by repeating ten-fold cross-validation ten times. Note that these cross-validation runs are performed *in addition* to the cross-validation performed during internal op-

Table 3.3: Accuracies for reduced-error pruning (REP) compared to significance pruning without (SIG) and with (SIG-SE) the one-standard-error rule

Dataset	REP	SIG	α_m	SIG-SE	α_m
anneal	98.5±0.4	98.5±0.4	1.0	98.3±0.4 •	0.1
audiology	72.8±1.5	72.2±1.5 •	1.0	70.0±1.2 •	0.1
australian	84.3±0.9	85.0±0.5 ∘	0.01	85.5±0.1 ∘	0.01
autos	63.3±2.8	63.2±2.9	1.0	61.9±2.9 •	0.4
balance-scale	78.4±0.9	78.3±1.0	1.0	77.2±1.4 •	0.3
breast-cancer	68.8±1.6	69.3±1.0	0.01	69.7±1.0	0.01
breast-w	94.3±0.7	94.2±0.6 •	0.3	93.6±0.7 •	0.05
german	72.4±1.4	72.1±1.3 •	0.4	71.2±1.6 •	0.2
glass-2	79.0±2.3	78.7±2.5	0.2	77.6±1.7 •	0.05
glass	66.6±3.4	66.3±3.8	1.0	64.6±3.6 •	0.05
heart-c	75.8±2.3	75.2±2.2 •	0.2	74.9±2.2	0.01
heart-h	78.0±1.7	78.4±1.6	0.05	78.2±1.5	0.01
heart-statlog	76.3±2.8	76.0±2.9	0.2	75.2±2.5	0.05
hepatitis	80.8±2.5	80.0±2.5	0.05	79.8±1.7	0.01
horse-colic	84.5±0.8	84.6±0.6	0.05	84.5±0.8	0.01
ionosphere	89.0±0.8	89.2±0.6	0.01	89.5±0.6	0.01
iris	94.6±0.8	94.7±0.5	0.01	94.7±0.5	0.01
labor	79.5±3.9	79.0±3.5	0.2	78.7±4.8	0.05
lymphography	75.2±1.7	74.6±1.3	0.2	73.8±1.2 •	0.05
pima-indians	74.0±0.6	73.9±0.8	0.2	73.5±0.9	0.01
primary-tumor	37.7±1.9	37.2±2.0 •	1.0	35.6±2.0 •	0.2
sonar	71.3±2.4	71.2±1.6	0.2	70.8±1.9	0.05
soybean	85.0±0.9	84.8±0.8	1.0	83.8±1.0 •	0.1
vehicle	71.0±1.1	70.9±1.5	1.0	70.0±1.4 •	0.2
vote	95.6±0.5	95.6±0.4	0.01	95.6±0.1	0.01
vowel	71.6±1.5	71.6±1.5	1.0	71.6±1.5	1.0
zoo	90.2±2.3	90.1±2.2	1.0	87.7±2.9 •	0.2

timization. The standard deviation for the ten data points corresponding to the ten cross-validation estimates are also shown in the tables. In Table 3.3, the ∘ symbol indicates for which datasets pruning based on significance testing produces significantly more accurate trees than reduced-error pruning. The • symbol indicates for which datasets it produces significantly less accurate trees. In Table 3.5, the ∘ symbol indicates for which datasets pruning based on significance testing produces significantly smaller trees than reduced-error pruning. A difference is considered to be significant if the corresponding cross-validation estimates are significantly different at the 0.05% level according to a paired two-sided t-test for the ten data points involved. Table 3.3 also shows the median significance levels chosen by SIG and SIG-SE across the 100 folds of the ten cross-validation runs, denoted by α_m.

Table 3.4: Results of paired t-tests (p=0.05) for accuracies: number indicates how often method in column significantly outperforms method in row

	REP	SIG	SIG-SE
REP	–	1	1
SIG	5	–	2
SIG-SE	13	12	–

The results of the significance tests for accuracy and size are summarized in Tables 3.4 and 3.6 respectively. In these tables, each entry indicates the number of datasets for which the method associated with its column significantly outperforms the method associated with its row.

Table 3.4 shows that pruning with significance testing (SIG) produces trees that are about as accurate as those for reduced-error pruning (REP) if the one-standard-error rule is not used. SIG is significantly more accurate than REP on one dataset (second column, first row) and significantly less accurate on five (first column, second row). However, the difference in accuracy is always very small. The largest difference occurs for the australian dataset (0.7%). If the one-standard-error rule is used (SIG-SE), the difference in performance grows larger. SIG-SE is significantly more accurate than REP on one dataset and significantly less accurate on thirteen. The results indicate that SIG-SE frequently overprunes: it produces decision trees that are too small and do not capture all the relevant information.

Table 3.6 shows how the methods compare with respect to the size of the pruned trees. Because SIG and SIG-SE can never prune less than REP, they never produce larger trees. On the other hand, SIG produces significantly smaller trees than REP for 24 datasets. For several datasets, for example, australian, breast-cancer, heart-h, hepatitis, pima-indians, and vote, the reduction in tree size is quite dramatic. SIG-SE produces even smaller trees than SIG—significantly smaller than REP on 26 datasets.

Taken together, these results mean that pruning with significance tests successfully improves on reduced-error pruning if the significance level is chosen according to the properties of the domain. The appropriate significance level can be identified automatically by cross-validation. The resulting trees are often much smaller and about as accurate. If the one-standard-error rule is used in conjunction with

Table 3.5: Tree sizes for reduced-error pruning (REP) compared to significance pruning without (SIG), and with (SIG-SE) one-standard-error rule

Dataset	REP	SIG		SIG-SE	
anneal	43.8±2.7	33.7±1.0	∘	33.2±0.9	∘
audiology	40.2±2.6	38.1±2.6	∘	31.0±1.4	∘
australian	34.3±7.9	10.1±2.7	∘	3.7±0.8	∘
autos	53.6±2.8	53.4±2.9		49.5±3.1	∘
balance-scale	54.6±3.2	51.9±3.7	∘	37.7±2.7	∘
breast-cancer	48.6±5.3	11.7±7.8	∘	2.3±1.7	∘
breast-w	15.1±1.5	13.3±1.9	∘	10.4±1.1	∘
german	95.8±9.5	72.4±10.1	∘	40.9±9.6	∘
glass-2	12.7±2.3	10.9±2.0	∘	8.3±1.6	∘
glass	23.4±1.8	22.1±1.8	∘	17.1±1.4	∘
heart-c	21.1±1.6	14.9±3.2	∘	8.8±1.7	∘
heart-h	15.3±1.8	8.2±1.3	∘	5.2±0.8	∘
heart-statlog	15.7±1.6	11.2±2.1	∘	7.6±1.4	∘
hepatitis	6.5±1.9	3.5±0.9	∘	2.0±0.4	∘
horse-colic	17.3±3.9	9.4±2.4	∘	6.7±0.6	∘
ionosphere	10.6±2.1	8.0±1.1	∘	6.4±1.0	∘
iris	6.4±0.5	5.8±0.4	∘	5.7±0.3	∘
labor	6.7±0.8	6.1±0.8	∘	4.9±1.0	∘
lymphography	14.6±3.0	11.8±3.1	∘	7.5±1.8	∘
pima-indians	35.8±5.4	21.7±3.7	∘	10.9±3.7	∘
primary-tumor	49.9±4.7	41.7±5.8	∘	23.5±5.1	∘
sonar	11.0±1.3	8.9±1.1	∘	5.6±1.0	∘
soybean	89.1±1.9	85.0±2.5	∘	72.9±2.7	∘
vehicle	65.5±4.5	59.4±5.2	∘	40.2±5.9	∘
vote	7.5±1.2	4.6±1.1	∘	3.4±0.6	∘
vowel	280.9±4.3	280.9±4.3		280.9±4.3	
zoo	14.1±0.4	14.0±0.4		13.1±0.5	∘

the cross-validation estimate, tree sizes can be reduced even further. However, this results in a noticeable loss of accuracy on some datasets.

3.6 Related work

Pruning methods for decision trees are one of the most extensively researched areas in machine learning. Several surveys of induction methods for decision trees have been published (Safavian & Landgrebe, 1991; Kalles, 1995; Breslow & Aha, 1997b; Murthy, 1998), and these all discuss different pruning strategies. In addition, empirical comparisons of a variety of different pruning methods have been conducted. This section first discusses the most popular pruning methods in the context of

Table 3.6: Results of paired t-tests (p=0.05) for sizes: number indicates how often method in column significantly outperforms method in row

	REP	SIG	SIG-SE
REP	–	24	26
SIG	0	–	25
SIG-SE	0	0	–

published experimental comparisons, highlighting their weaknesses as well as proposed remedies. Then we summarize prior work on the problem of underpruning, of which a particular instance is tackled in this chapter. Finally, we briefly discuss less well known pruning techniques and modifications to existing procedures.

Quinlan (1987a) was the first to perform a comparison of pruning methods. He presents experimental results for three methods: cost-complexity pruning, reduced-error pruning, and pessimistic error pruning. Another experimental comparison of several pruning methods has been performed by Mingers (1989). As well as the three investigated by Quinlan, Mingers includes two more procedures in his comparison: critical value pruning and minimum-error pruning. However, his comparison has been criticized because he does not give all pruning methods access to the same amount of data. Also, he uses a non-standard version of reduced-error pruning. The critics, Esposito et al. (1997), compared essentially the same pruning algorithms, but in order to make a fair comparison, their experimental procedure assures that all algorithms have access to the same amount of data when generating the pruned tree. In contrast to Mingers, they use Quinlan's original version of reduced-error pruning, as well as a more recent incarnation of minimum-error pruning. Their paper includes results for a successor of pessimistic error pruning called error-based pruning.

3.6.1 Cost-complexity pruning

Cost-complexity pruning was introduced in the classic CART system for inducing decision trees (Breiman et al., 1984). It is based on the idea of first pruning those subtrees that, relative to their size, lead to the smallest increase in error on the training data. The increase in error is measured by a quantity α that is defined to be the average increase in error per leaf of the subtree. CART uses α to generate a sequence of increasingly smaller pruned trees: in each iteration, it prunes all subtrees that exhibit the smallest value for α. Each tree corresponds to one particular value α_i. In order to choose the most predictive tree in this sequence, CART either uses a hold-out set to estimate the error rate, or employs cross-validation. Cross-validation poses the additional problem of relating the α_j^k values observed in training fold k to the α_i values from the original sequence of trees, for these values are usually different. CART solves this problem by computing the geometric average α_i^{av} of α_i and α_{i+1} for tree i from the original sequence. Then, for each fold k of the cross-validation, it picks the tree that exhibits the largest α_j^k value smaller than α_i^{av}. The average of the error estimates for these trees is the cross-validation estimate for tree i.

Discussion

It can be shown that the sequence of pruned trees can be generated in time that is quadratic in the number of nodes (Esposito et al., 1997). This is significantly slower than the pruning methods investigated in this chapter, which are linear in the number of nodes. It implies that CART's runtime is quadratic in the number of training instances if the number of nodes increases linearly with the number of training instances—a realistic scenario in noisy real-world datasets. Note that, as well as allowing for cross-validated error estimates, CART also introduced the one-standard-error rule discussed in Section 3.5.

Quinlan (1987a, page 225) notes that it is unclear why the particular cost-complexity model used by CART "... is superior to other possible models such as the product of the error rate and number of leaves," and he also finds that "... it seems anomalous that the cost-complexity model ... is abandoned when the best tree is selected." Consequently he introduces two new pruning methods: reduced-error pruning, which has been discussed above, and pessimistic error pruning.

3.6.2 Pessimistic error pruning

Pessimistic error pruning is based on error estimates derived from the training data. Hence it does not require a separate pruning set. More specifically, pessimistic error pruning adds a constant to the training error of a subtree by assuming that each leaf automatically classifies a certain fraction of an instance incorrectly. This fraction is taken to be 1/2 divided by the total number of instances covered by the leaf, and is called a "continuity correction" in statistics (Wild & Weber, 1995). In that context it is used to make the normal distribution more closely approximate the binomial distribution in the small sample case. During pruning this adjustment is used in conjunction with the one-standard-error rule above. A tree is made into a leaf if the adjusted error estimate for the leaf is smaller or equal to the adjusted error of the tree plus one standard error of the latter estimate. The standard error is computed by assuming that the adjusted error is binomially distributed. A subtree is considered for replacement *before* its branches are pruned.

Discussion

In contrast to reduced-error pruning, which proceeds in a bottom-up fashion, pessimistic error pruning uses the top-down approach and considers pruning a tree before it prunes its subtrees. Hence it is marginally faster in practice. However, its worst-case time complexity is also linear in the number of nodes in the unpruned tree. As Breslow and Aha (1997b) note, top-down pruning methods suffer from the "horizon effect": a tree might be pruned even when it contains a subtree that "... would not have been pruned by the same criterion."

In his experiments comparing cost-complexity pruning, reduced-error pruning, and pessimistic error pruning, Quinlan (1987a) finds that cost-complexity pruning tends to over-prune: it generates the smallest trees but they are slightly less accurate than those produced by the other two methods. He concludes that pessimistic error pruning is the preferred method since it does not require a separate pruning set.

Buntine (1992) also compares pessimistic pruning, and variants of cost-complexity pruning. However, none of these variants seems to implement the version of cost-complexity pruning as defined by Breiman et al. (1984): Buntine uses the *pruning* data to compute the cost-complexity of a tree for a given α instead of the *training* data used in the original formulation. In his terminology, the pruning data is called the "test set," and he claims that "The substitution error estimate is usually computed on a test set" (page 93). He later acknowledges that "The most likely place for bugs in the existing implementation [of the tree learner used in his experiments] is in the cost complexity pruning module. . . " (page 99). Buntine's implementation is likely to produce overly complex decision trees because it gives the algorithm a chance to fit the pruning data before the final pruned tree is selected.

3.6.3 Critical value pruning

Critical value pruning (Mingers, 1987) is a bottom-up technique like reduced-error pruning. However, it makes pruning decisions in a fundamentally different way. Whereas reduced-error pruning uses the estimated error on the pruning data to judge the quality of a subtree, critical value pruning looks at information collected during tree growth. Recall that a top-down decision tree inducer recursively employs a selection criterion to split the training data into increasingly smaller and purer subsets. At each node it splits in a way that maximizes the value of the splitting criterion—for example, the information gain. Critical value pruning uses this value to make pruning decisions. When a subtree is considered for pruning, the value of the splitting criterion at the corresponding node is compared to a fixed threshold, and the tree is replaced by a leaf if the value is too small. However, one additional constraint is imposed: if the subtree contains at least one node whose value is greater than the threshold, it will not be pruned. This means that a subtree is only considered for pruning if all its successors are leaf nodes.

Discussion

The performance of critical value pruning depends on the threshold used for the pruning decisions. Assuming that the splitting criterion's value increases with the quality of the split, larger thresholds result in more aggressive pruning. Of course, the best value is domain-dependent and must be found using a hold-out set or cross-validation. The main difference between critical value pruning and other post-pruning methods is that it looks solely at the information provided by the splitting criterion, only indirectly taking account of the accuracy of the subtree considered for pruning. Therefore, its performance depends critically on how well the splitting criterion predicts the subtree's generalization performance. Most splitting criteria, however, do not take account of the number of instances that support the subtree. Consequently the procedure overestimates the quality of subtrees that cover only a few instances. Among those splitting criteria used by Mingers, only the chi-squared distribution is an exception. However, because it is used to grow the tree, the derived probabilities are highly biased (Jensen & Schmill, 1997).

3.6.4 Minimum-error pruning

Minimum-error pruning was invented by Niblett and Bratko (1987). It is similar to pessimistic error pruning in that it uses class counts derived from the training data. However, it differs in the way it adjusts these counts in order to more closely reflect a leaf's generalization performance. In its initial version, which was used by Mingers (1989), the adjustment is a straightforward instantiation of the Laplace correction, which simply adds one to the number of instances of each class when the error rate is computed. Like reduced-error pruning, minimum-error pruning proceeds in a bottom-up fashion, replacing a subtree by a leaf if the estimated error for the former is no smaller than for the latter. In order to derive an estimate of the error rate for a subtree, an average of the error estimates is computed for its branches, weighted according to the number of instances that reach each of them.

Discussion

In a later version of minimum-error pruning, Cestnik and Bratko (1991) refine the Laplace heuristic. Instead of adding one to the count for each class, they add a constant $p_i \times m$, where p_i is the class' prior probability in the training data and m is a factor that determines the severity of the pruning process. Higher values for m generally produce smaller trees because they reduce the influence of the training data and result in a "smoothing" effect that tends to equalize the probability estimates at different leaves. However, a higher value does not automatically produce a smaller tree (Esposito et al., 1997). This is a significant disadvantage because it means that for each value of m considered, the procedure must begin with an unpruned tree. Since the best value for m can only be found by estimating the error of the pruned tree on a hold-out set, or using cross-validation, this property makes minimum-error pruning considerably slower than reduced-error pruning.

Mingers (1989) compares the five pruning methods discussed above. In his experiments, those methods that use a separate pruning set outperform ones that are just based on estimates from the training data. However, the comparison has been criticized for the experimental methodology employed (Esposito et al., 1997). In Mingers' experiments, methods with a separate pruning set for parameter selection or error estimation have an unfair advantage because the pruning data is provided *in addition* to the training data used for the other methods. His experimental results for critical value pruning, error-complexity pruning and reduced-error pruning on the one side, and minimum-error pruning and pessimistic error pruning on the other side, can therefore not be compared directly.

However, his results *can* be used to compare the performance of the methods within each group (Breslow & Aha, 1997b). They show, for example, that minimum-error pruning produces less accurate and larger trees than pessimistic error pruning. They also show that cost-complexity pruning produces smaller trees than reduced-error pruning with similar accuracy. However, this latter result is questionable because Mingers' version of reduced-error pruning differs from the original algorithm proposed by Quinlan (1987a). In Mingers' experiments, critical value pruning is less accurate than both cost-complexity and reduced-error pruning, and it produces larger trees than cost-complexity pruning. Mingers also investigated whether different splitting criteria and pruning methods interact, and found that this is not the case. This means that these two components of a decision tree inducer can be studied independently (Breslow & Aha, 1997b).

3.6.5 Error-based pruning

Error-based pruning is the strategy implemented by the well-known decision tree inducer C4.5 (Quinlan, 1992). A similar strategy has also been proposed by Kalkanis (1993). Like pessimistic error pruning, it derives error estimates from the training data, assuming that the errors are binomially distributed. However, instead of the one-standard-error rule employed by pessimistic error pruning, it computes a confidence interval on the error counts based on the fact that the binomial distribution is closely approximated by the normal distribution in the large sample case. Then, the upper limit of this confidence interval is used to estimate a leaf's error rate on fresh data. In C4.5, the confidence interval is set to 25% by default. Like reduced-error pruning—and in contrast with pessimistic error pruning—a bottom-up traversal strategy is employed: a subtree is considered for replacement by a leaf after all its branches have already been considered for pruning. Replacement is performed if the error estimate for the prospective leaf is no greater than the sum of the error estimates for the current leaf nodes of the subtree. As well as subtree replacement, C4.5 also performs a pruning operation called "subtree raising" that replaces a subtree with its most populated branch if this does not increase the estimated error.

Discussion

Using a confidence interval is a heuristic way of reducing the optimistic bias in the error estimate derived from the training data, but it is not statistically sound, and Quinlan (1992) acknowledges this fact. From a statistical perspective this pruning procedure shares the problems of pessimistic error pruning. The use of the normality assumption is also questionable because it is only correct in the limit. For small samples with less than 100 instances, statisticians use the Student's distribution instead of the normal distribution (Wild & Weber, 1995). In decision tree induction, small samples are exactly those which are most likely to be relevant in the pruning process.

Esposito et al. (1997) claim that error-based pruning and pessimistic error pruning behave in the same way. However, this assertion seems to conflict with the experimental results presented in their paper. For all but one of 15 datasets investigated, pessimistic error pruning produces significantly smaller trees than error-based pruning (see Table 8 in their paper). With respect to accuracy, however, the differences between the two methods are only minor. Considering these results, it is not clear why error-based pruning has replaced pessimistic error pruning in C4.5.

In contrast to Mingers (1989), Esposito et al. (1997) do not find that methods operating with a pruning set produce more accurate trees than those that rely solely on the training data. As mentioned above, this difference is due to the particular experimental procedure that Mingers employs.

Esposito et al. (1997) also introduce the notion of an "optimally pruned tree." This tree is derived by applying reduced-error pruning using the *test set* as the pruning data. The authors claim that it is possible to determine if a particular method "overprunes" or "underprunes" by comparing its tree size to the size of the optimally pruned tree. However, this neglects the fact that reduced-error pruning often overfits the pruning data. As the results from this chapter show, their procedure is likely to detect overpruning when in fact even more pruning is warranted. However, it is safe to say that it correctly detects underpruning, and it shows that minimum-error pruning, critical value pruning, and error-based pruning generally produce trees that are too large.

Breslow and Aha (1997b) summarize the main results from Mingers (1987) and Esposito et al. (1997) by showing differences and similarities in their findings. They conclude that pessimistic error pruning and error-based pruning produce the most accurate trees among the methods compared. However, this claim seems too general. The results from Esposito et al. (1997) show that these two pruning methods are only superior when little pruning is warranted; in several cases they produce less accurate trees than, for example, reduced-error pruning. This is a direct result of their tendency to underprune, which is also mentioned by Breslow and Aha (1997b). They conclude that "... these findings on post-pruning algorithms are preliminary ..." and that "... further investigation is needed ..."

The same authors (Breslow & Aha, 1997a) have also published an empirical comparison of "tree-simplification procedures" that includes two pruning methods: error-based pruning in Revision 8 of C4.5 (Quinlan, 1996), and ITI's pruning procedure (Utgoff et al., 1997) that is based on the minimum description length principle (Quinlan & Rivest, 1989). They found that C4.5 generally performed best. They also performed an experiment in which they tuned C4.5's pruning parameters using nine-fold cross-validation. Unfortunately they compare these results to those of an *unpruned* tree. Consequently it is difficult to judge whether parameter tuning improves on the default parameter settings. Kohavi (1995b, page 122) also uses cross-validation to choose parameter settings for C4.5. He reports that it is "... hard to judge whether C4.5 with automatic parameter tuning ... is significantly superior to C4.5." However, he only looks at the accuracy of the resulting trees, not their size.

3.6.6 Underpruning

Oates and Jensen (1997) were the first to observe that bottom-up procedures like error-based pruning and reduced-error pruning can produce overly large decision trees. They find that the size of pruned decision trees can be reduced significantly, with only a small loss in accuracy, if a random subsample of the original training data is passed to the induction algorithm instead of the full dataset. This implies that the pruning process does not simplify the tree sufficiently when more data becomes available. Later, they argue that the reason for this overfitting is a phenomenon they call "error propagation" (Oates & Jensen, 1998). Error propagation is due to to the bottom-up fashion in which pruning proceeds: a subtree is considered for replacement by a leaf *after* its branches have been considered for pruning. Recall that a subtree is only retained if it has lower estimated error than the corresponding leaf. Because the subtree has already been modified before the error estimate is computed, it is optimistically biased, rendering it less likely to be pruned—spurious correlations in the lower parts of the subtree are propagated to its root node, making it unduly likely that it will be retained. Of course, the deeper the subtree, the more opportunities there are for these spurious associations to occur, and the more likely it is that the subtree will survive. Oates and Jensen (1999) show that it is possible to quantify the survival probability in an approximate way by making some simplifying assumptions. They also propose two ways of preventing overly complex decision trees from being built. Note that this chapter shows how standard significance tests can be used to detect spurious correlations, thereby minimizing the effect of error propagation in reduced-error pruning.

Randomization pruning

The first method proposed by Oates and Jensen (Oates & Jensen, 1998), called "randomization pruning," is based on the idea of a permutation test, and is applied as a post-processing step to simplify the pruned tree. For each subtree of the pruned tree, the probability $\hat{p}$ that an equally or more accurate subtree would have been generated *by chance alone* is computed. To do this, the procedure collects the training data from which the subtree was built, and randomly permutes its class labels. Then it applies the decision tree inducer to this randomized data, and records the accuracy of the resulting tree. This randomization procedure is repeated N times. The probability $\hat{p}$ is the fraction of times for which the randomized subtrees are no less accurate than the original subtree. If $\hat{p}$ is greater than a certain threshold—the authors suggest 0.05—the subtree is discarded and replaced by a leaf node. In the three datasets investigated by the authors, this procedure successfully reduces the size of the pruned trees built by error-based pruning, and slightly reduces accuracy in only one of them.

This randomization procedure applies a significance test just like the methods investigated in this chapter. However, it differs in that it is computationally very expensive. The induction algorithm is applied N times for each subtree in the original pruned tree—and in order to obtain accurate probability estimates, the required value for N can be anywhere between several hundred and several thousand. Therefore this procedure is of theoretical rather than practical value. There is also a conceptual problem with randomization pruning. Consider the hypothetical situation where it is used in conjunction with a tree inducer that does not perform any pruning at all. If all training instances are unique—a scenario that is likely to occur in practice—an unpruned tree will never be significant because it will be 100% accurate on every randomized version of the training data. Thus $\hat{p}$ will be one and the tree will be pruned back to its root node, regardless of how predictive it is. This means that the effectiveness of randomization pruning depends on the effectiveness of the underlying pruning procedure.

Reduced-overlap pruning

The second approach proposed by Oates and Jensen (1999) is designed to improve reduced-error pruning. It is based on the idea that unbiased error estimates for each subtree can be obtained if a fresh set of pruning data is available to evaluate each of them. Using artificial datasets, where unlimited amounts of new data can be generated, the authors show that this successfully prevents reduced-error pruning from building overly complex decision trees. In practice, however, only a limited amount of pruning data is available, and so it is necessary to use random subsamples of the original pruning data instead of fresh data to derive error estimates at each node of the tree. The idea is to minimize the overlap between these random subsamples in order to minimize dependencies between the error estimates. Using random subsamples, each containing 50% of the original pruning data, Oates and Jensen find that this method leads to significantly smaller trees for 16 out of 19 practical datasets investigated, and significantly decreases accuracy on only one of them.

However, it is plausible that the reduction in tree size is solely due to the smaller amount of pruning data that is used at each node, and does not result from the increased independence between the samples. Note also that reduced-overlap pruning does not take sampling variance into account. Thus, although it reduces error propagation, it can still overfit the pruning data if chance associations occur. We repeated Oates and Jensen's experiment using the datasets and the experimental setting from the last section, and found that it significantly decreased accuracy for 12 of the 27 datasets and produced significantly smaller trees for all 27. The results are summarized in Appendix A.2. On most of the 12 datasets where accuracy degraded significantly, it decreased by several percent—for example, by more than 8% on the vowel dataset. This indicates that reduced-overlap pruning prunes too aggressively on these datasets.

3.6.7 Other pruning methods

Apart from the pruning algorithms discussed above, several less well-known methods have been proposed in the literature that are either modifications of existing algorithms or based on similar ideas.

Crawford (1989) uses cost-complexity pruning in conjunction with the .632 bootstrap (Efron & Tibshirani, 1993) for error estimation, substituting it for the standard cross-validation procedure. However, Weiss and Indurkhya (1994a,b) demonstrate that cross-validation is almost unbiased and close to optimal in choosing the right tree size. Kohavi (1995b) shows that the .632 bootstrap has higher bias but lower variance than cross-validation, noting that it can be preferable for small sample sizes. Later, Efron and Tibshirani (1997) proposed an improved bootstrap estimator, the .632+ bootstrap, with lower bias. Gelfand et al. (1991) modify CART's pruning procedure by interleaving the growing and pruning phases: a tree is grown using one half of the data, then pruned using the other half. In subsequent iterations, the existing tree continues to be modified by these two steps, but in each iteration the roles of pruning and growing data are exchanged. According to results presented by Gelfand et al., this procedure speeds up the pruning process and produces more accurate trees. However, the trees are also larger.

Minimum description length principle

Several authors have proposed pruning methods based on the minimum description length (MDL) principle (Rissanen, 1978). These methods derive from the idea that a successful inducer will produce a classifier that *compresses* the data, and exploit the fact that the complexity of a model, as well as the complexity of a dataset, can be measured in "bits" given an appropriate coding scheme. Induction is considered to be successful if the cost of coding both the classifier, and its classification errors, is lower than the cost of coding the training data itself. Moreover, the greater the reduction in coding cost (the compression), the "better" the inducer. MDL pruning algorithms seek decision trees that maximally compress the data. They differ in the coding scheme they employ. Successful application of the MDL principle depends on how well the coding scheme matches the particular properties of the learning problem at hand. This is a direct consequence of the fact that it is a reformulation of Bayes' rule (Buntine, 1992), in which probabilities have been replaced by their logarithms. The prior probability in Bayes' rule determines the model's coding cost, and it is essential that the distribution of the prior probabilities is chosen appropriately. In the case of decision trees, for example, different prior distributions result in different amounts of pruning. Proponents of the MDL principle claim that it has two advantages: no parameters need to be chosen, and no pruning data needs to be set aside. However, they omit to mention that the choice of the prior distribution is a parameter in itself, and one can argue that it is a disadvantage that this parameter cannot be freely adjusted.

Quinlan and Rivest (1989) were the first to use the MDL principle for pruning decision trees. They compare it experimentally to pessimistic error pruning and obtain mixed results. Wallace and Patrick (1993) point out flaws in their coding scheme, but acknowledge that these flaws do not affect the outcome. Forsyth (1994) also uses an MDL approach, as do Mehta et al. (1995). The latter authors report that their method produces smaller trees than both pessimistic error pruning and error-based pruning, but larger trees than cost-complexity pruning. Error rates are similar in each case.

Optimal pruning

Another line of research investigates "optimal" pruning algorithms that produce a sequence of smaller and smaller pruned trees, where each tree has the property that it is the most accurate one on the training data among all pruned trees of the same size. Breiman et al. (1984) were the first to suggest a dynamic programming solution to this problem, and Bohanec and Bratko (1994) present a corresponding algorithm. They note that cost-complexity pruning, discussed above, produces a sequence of optimal trees that is a subset of the sequence generated by their method. The worst-case time complexity of their algorithm is quadratic in the number of leaves of the unpruned tree. Almuallim (1996) presents an improved optimal pruning algorithm, also based on dynamic programming, that has slightly lower worst-case time complexity. Neither method addresses the question of how to choose tree size in order to maximize generalization performance. Bohanec and Bratko suggest that this can be done using cross-validation, but do not test this experimentally.

Pruning with costs

Often, real-world learning problems involve costs because some classification errors are more expensive than others. The literature contains several approaches to cost-sensitive pruning (Knoll et al., 1994; Bradford et al., 1998; Vadera & Nechab, 1994). Ting (1998) presents an elegant solution for incorporating costs that is based solely on weighting the training instances. By resampling instances with a probability proportional to their weight, his methods can be applied to learning schemes that cannot make use of weights directly.

Pruning with significance tests

Statistical significance tests have been applied to learning algorithms before, but in almost all cases using information derived during training—a procedure that is questionable if appropriate adjustments are not performed (Cohen & Jensen, 1997). In Quinlan's ID3 decision tree inducer (1986b), for example, the parametric chi-squared test is used to decide when to stop splitting the training data into increasingly smaller subsets. The same technique is also used by the decision tree inducer CHAID (Kass, 1980). These pre-pruning techniques will be discussed in more detail in Chapter 4.

Jensen et al. (1997) apply critical value pruning in conjunction with the chi-squared distribution. However, instead of using the probabilities directly—which is incorrect because they have been used for training—they apply a statistical technique known as the "Bonferroni correction" to make an appropriate adjustment. Statisticians use the Bonferroni correction to adjust the significance levels of multiple statistical hypothesis tests (Westfall & Young, 1993). It requires that the p-values of the tests are independent, which is unlikely to be true in real-world learning situations (Jensen, 1992). If the independence assumption is not fulfilled, a permutation test on the original test's p-values is the statistically correct way of adjusting for multiple comparisons (Westfall & Young, 1993). In order to apply the Bonferroni adjustment, it is necessary to know *in advance* how many significance tests will be performed. However, because pruning is a dynamic process, this knowledge is impossible to achieve *a priori*. Despite these theoretical problems, Jensen and Schmill (1997) report good results for their method in practice: it often produces smaller trees than C4.5 and is seldom less accurate.

Some authors have investigated the use of permutation tests in learning algorithms. Gaines (1989) uses the one-sided version of Fisher's exact test to evaluate classification rules, and employs the Bonferroni correction to adjust for multiple tests. Jensen (1992) proposes a *conditional* permutation test based on the chi-squared statistic to decide when to modify and expand rules in a prototypical rule learner. In order to perform the significance test, he uses fresh data that has not been used for model fitting. Li and Dubes (1986) propose a version of Fisher's exact test for attribute selection and pre-pruning in binary domains. Frank and Witten (1998b) use the more general Freeman and Halton test for the same purpose, and find that post-pruning with error-based pruning performs better. Martin (1997) proposes using the *test statistic* of the Freeman and Halton test for attribute selection and pre-pruning; however, results improve when the full significance test is used instead (Frank & Witten, 1998b). Hong et al. (1996) compute the expected value of the test statistic under the permutation distribution and use this to normalize the value from the original data. They propose to use this normalized value for attribute selection and pre-pruning.

Computational learning theory

There has also been some work in computational learning theory on post-pruning algorithms for decision trees. Kearns and Mansour (1998) extend earlier work by Mansour (1997), and present a bottom-up algorithm similar to C4.5's error-based pruning that produces a near-optimum pruning such that—in a theoretical sense—its generalization error is almost as low as the one for the hypothetical best pruning of the tree. However, the experimental results presented by Mansour (1997) show that there is little difference between the two methods in practice.

3.7 Conclusions

This chapter investigates whether standard significance tests can be used to improve the reduced-error pruning algorithm to make the pruned decision trees smaller and at least as accurate. The experimental results show that this is indeed the case, if the tests' significance levels are adjusted according to the amount of pruning required by the domain. Thus the primary hypothesis of this chapter is correct. The experimental results also show that an appropriate significance level can be found automatically using cross-validation.

Experiments comparing the performance of several permutation tests and the parametric chi-squared test show that they produce trees of different sizes for a given significance level. However, the differences can be eliminated by tuning the significance level for each test individually. Hence the secondary hypothesis of this chapter turns out to be incorrect: in practice the parametric test and the permutation tests produce pruned trees with very similar size and accuracy if the significance levels for each test are chosen appropriately. Therefore, since the permutation tests are computationally more expensive than the parametric test, there is no reason to use them in this particular application.

For a fixed significance level, the additional computational complexity incurred by a significance test is negligible when the parametric test is employed. However, for best results the significance level needs to be chosen via cross-validation or a hold-out set. Cross-validation, which is the preferred method for small datasets, increases the run time by a constant factor. The fully expanded tree needs to be generated only once for each fold of the cross-validation. Standard reduced-error pruning also needs to be done only once. For every significance level, pruning with significance testing can start with the tree pruned by reduced-error pruning.

In time-critical applications, where the classifier's perspicuity is not an issue, there is often no advantage in using significance tests in addition to the standard reduced-error pruning procedure. However, when comprehensibility is important, the extra time for the cross-validation is well spent.

Chapter 4

Pre-pruning

Top-down induction of decision trees is arguably the most popular learning regime in classification because it is fast and produces comprehensible output. However, the accuracy and size of a decision tree depends strongly on the pruning strategy employed by the induction algorithm that is used to form it.

Pruning algorithms discard branches of a tree that do not improve accuracy. To achieve this they implement one of two general paradigms: pre-pruning or post-pruning. Pre-pruning algorithms do not literally perform "pruning" because they never prune existing branches of a decision tree: they "prune" in advance by suppressing the growth of a branch if additional structure is not expected to increase accuracy. Post-pruning methods, discussed in the previous chapter, have their equivalent in horticulture: they take a fully grown tree and cut off all the superfluous branches—branches that do not improve predictive performance.

In the machine learning community, folklore has it that post-pruning is preferable to pre-pruning because of "interaction effects": in univariate decision trees, pre-pruning suppresses growth by evaluating each attribute individually, and so might overlook effects that are due to the interaction of several attributes and stop too early. Post-pruning, on the other hand, avoids this problem because interaction effects are visible in the fully grown tree. A more general version of the phenomenon is known as the "horizon effect": the stopping criterion in a pre-pruning method can prune a branch leading to a subtree that would not be pruned according to the same criterion.

A simple example nicely illustrates the influence of interaction effects. Consider the parity problem discussed in the last chapter. In this problem, the (binary) class is 1 if and only if the number of (binary) attributes with value 1 is even. Hence—if all attribute values are uniformly distributed—*no* single attribute has any predictive power: *all* attribute values must be known to determine the class. Given any one particular attribute—or even any subset of attributes—both classes look equally likely. In other words, only a fully expanded decision tree shows that the parity data contains any information. Post-pruning procedures have access to the unpruned tree and are therefore able to retain fully expanded branches. Pre-pruning methods, on the other hand, are necessarily inferior: since no single attribute exhibits any correlation with the class, they produce a decision tree consisting of a root node only. Consequently pre-pruning methods, based chiefly on the parametric chi-squared test, have been abandoned in favor of various post-pruning strategies.

However, it is unclear how frequently parity problems occur in real-world domains. It is plausible that, in practical datasets, every relevant attribute is strongly correlated with the class. In fact, many methods for attribute selection are based on this assumption. Moreover, people who want to use machine learning techniques for classification usually try to come up with attributes that are predictive of the class. If this is the case, there is no obvious reason to believe that pre-pruning is inferior.

Given these observations it is plausible that pre-pruning is competitive with post-pruning on practical learning tasks. This chapter introduces a sequence of refinements to the standard pre-pruning procedure and verifies empirically whether they improve its performance. It also compares pre-pruning with post-pruning on practical benchmark datasets. In this comparison, both procedures are implemented using the same statistical criteria to allow a fair comparison.

As discussed in the previous chapter, the parametric chi-squared test, traditionally used as the stopping criterion in pre-pruning procedures, relies on large-sample approximations that do not necessarily hold when applied to the small samples occuring at the nodes of a decision tree. The first hypothesis of this chapter is that a permutation test based on the chi-squared statistic improves on the parametric chi-squared test when used for pre-pruning decision trees.

Hypothesis 4.1. *If tree A is pre-pruned using the parametric chi-squared test, and tree B is pre-pruned using a corresponding permutation test, and both trees have the same size, then A will be less accurate than B.*

Note that, when pre-pruning, the significance test is used to determine whether there is a significant association between the values of an attribute and the class values. In the previous chapter it is used to test whether a subtree's predictions and the class values in the pruning set are correlated.

At each node of the decision tree, pre-pruning methods apply a significance test to every attribute being considered for splitting. Splitting ceases if no significant attribute can be found. This means that several significance tests are performed at each node. However, multiple significance tests increase the probability that a significant association is found just by chance: the more attributes are tested, the more likely it becomes that the test outputs a low p-value for one of them. The reason for this is that the p-value of a significance test is itself a random variable. The more values of this variable are observed, the more likely it is that an unusually low one will be encountered. In other words, the more attributes there are, the more likely it is that the pre-pruning procedure will continue splitting—even when additional attributes do not make the predictions more accurate.

Statisticians have devised various methods to adjust the significance level appropriately in situations with multiple significance tests. This chapter investigates whether one of them, known as the "Bonferroni correction," improves the performance of significance tests in pre-pruning procedures.

Hypothesis 4.2. *If tree A is pre-pruned without adjusting for multiple significance tests, and tree B is pre-pruned by performing appropriate adjustments, and both trees have the same size, then A will be less accurate than B.*

As mentioned above, it is commonly assumed that pre-pruning is inferior to post-pruning. However, published empirical comparisons are rare. Moreover, the few that do exist use different statistical techniques for implementing the two strategies, which makes it difficult to say whether the difference in performance is due to the particular techniques employed or to the adoption of either of the two pruning paradigms.

This chapter includes an experiment that compares the two paradigms on a level footing. It uses the same basic statistical techniques for both pre- and post-pruning.

Hypothesis 4.3. *If tree A is pre-pruned, and tree B post-pruned using the same statistical techniques, and both trees have the same size, then A will be less accurate than B.*

In statistical terms, this hypothesis states that post-pruning is more *powerful* a test than pre-pruning. This is plausible because post-pruning has access to more precise statistics about the data than pre-pruning, statistics that have been uncovered by building the full decision tree. This additional information gives it a competitive edge.

The structure of this chapter is as follows. Section 4.1 explains how significance tests can be used for pre-pruning and discusses the modifications to the standard procedure that are mentioned above. Section 4.2 describes experiments that test the usefulness of these modifications on practical datasets. It also compares pre- and post-pruning. Section 4.3 discusses related work, and Section 4.4 summarizes the findings of this chapter.

4.1 Pre-pruning with significance tests

In decision trees, pre-pruning is essentially a problem of attribute selection: given a set of attributes, find those ones that are predictive of the class. In contrast to *global* attribute selection, which eliminates irrelevant attributes in a dataset prior to induction, pre-pruning uses *local* attribute selection: at each node of a decision tree, it tries to find the attributes that are relevant in the local context. If no relevant attributes can be found, splitting stops, and the current node is not expanded any further. Otherwise, one of the selected attributes is chosen for splitting.

A predictive attribute exhibits a significant association between its values and the class values. Thus finding a predictive attribute is a matter of determining whether it is correlated to the class, and whether this correlation is significant. This is exactly the kind of situation that statistical significance tests are designed for. Hence they are obvious candidates for pre-pruning criteria. A generic pre-pruning algorithm based on significance tests is depicted in Figure 4.1. For simplicity, we assume that all attributes are nominal and that the inducer extends one branch for each value of the nominal attribute.

At each node of a decision tree we must decide which attribute to split on. This is done in two steps. First, attributes are rejected if they show no significant association to the class according to a pre-specified significance level. Second, from the attributes that remain, the one with the lowest value of the splitting criterion

```
Procedure BuildTree(A: attributes, D: instances)

S := attributes from A that are significant at level α
if S is empty return
a := attribute from S that maximizes splitting criterion
for each value V_i of a
   BuildTree(A without a, instances from D with value V_i for a)
return
```

Figure 4.1: Generic pre-pruning algorithm

is chosen. This splitting criterion can be the attribute's information gain, but other criteria are also possible (Mingers, 1988). The selected attribute is then used to split the set of instances, and the algorithm recurses. If no significant attributes are found in the first step, the splitting process stops and the subtree is not expanded any further.

This gives an elegant, uniform, technique for pre-pruning that is a generic version of the procedure proposed by Quinlan (1986b), who uses the parametric chi-squared test for significance testing and the information gain criterion as the splitting criterion. The division into two steps is motivated by the two different statistical concepts of *significance* and *strength* (Press et al., 1988, p. 628). We test the significance of an association using a statistical test before we consider its strength as measured by the splitting criterion.

4.1.1 Significance tests on attributes

For attribute selection, we seek to test whether there is a significant association between an attribute's values and the class values. With I attribute values and J class values, this equates to testing for independence in the corresponding $I \times J$ contingency table. Appropriate significance tests for this purpose are discussed in the previous chapter, and are divided into parametric and non-parametric tests.

Quinlan (1986b) employs the parametric chi-squared test, which is based on the chi-squared distribution. As explained in the previous chapter, the chi-squared distribution is only an approximation to the true sampling distribution of the χ^2 statistic. With small samples and/or skewed distributions, this approximation is not accurate. Figure 4.2 shows the contingency tables for two example datasets. The chi-squared approximation is accurate for the table on the right because sufficient

Datasets

	3 instances	60 instances
Attribute values:	+ − −	− +
Class values:	A B A	B A ...

Contingency tables

	+	−	
A	1	1	2
B	0	1	1
	1	2	3

	+	−	
A	20	6	26
B	5	29	34
	25	35	60

Parametric chi-squared test valid?

3 instances	60 instances
no	yes

Figure 4.2: Example contingency tables for attribute selection

Attribute	Class	Permutations					
+	A	A_1	A_3	B_2	B_2	A_1	A_3
−	B	B_2	B_2	A_1	A_3	A_3	A_1
−	A	A_3	A_1	A_3	A_1	B_2	B_2
		a	b	c	d	e	f

Figure 4.3: Permutations of a small dataset

quantities of data are present. For the table on the left, however, this is not the case. The parametric chi-squared test is statistically valid as a stopping criterion when used near the root of a decision tree, where sufficient quantities of data are available. From a statistical point of view it is questionable to use it near the leaves, where data is sparse.

Fortunately there is an alternative, discussed in the previous chapter, that does apply in small frequency domains. A permutation test based on the the χ^2 statistic is statistically valid for any sample size because it computes the sampling distribution

a, b, e, f

	+	−	
A	1	1	2
B	0	1	1
	1	2	3

$\chi^2 = 0.75$

c, d

	+	−	
A	0	2	2
B	1	0	1
	1	2	3

$\chi^2 = 3$

Figure 4.4: Contingency tables for the permutations

Table 4.1: Average probabilities for random data (20 instances; non-uniformly distributed attribute values)

Attribute Values	Class Values	$\hat{p}_{\chi}$	p_{χ}
2	2	0.745	0.515
2	5	0.674	0.466
2	10	0.741	0.446
5	2	0.549	0.444
5	5	0.561	0.448
5	10	0.632	0.418
10	2	0.548	0.430
10	5	0.581	0.425
10	10	0.639	0.382

directly by enumerating all possible permutations of the data. Figure 4.3 shows all six permutations for the small dataset from Figure 4.2. Each of the permutations corresponds to a hypothetical contingency table. In this case, there are only two possible tables, depicted in Figure 4.4 together with their χ^2 values. The p-value of the test is the fraction of permutations for which the χ^2 statistic is at least as large as the value observed for the original class values. In this simple example, the p-value is 1 because, according to Equation 3.1 in Section 3.2.2, the χ^2 value for the original data is 0.75—the minimum value that can be achieved by any permutation of the data. In other words, the observed association between the attribute and the class is insignificant.

A simple experiment confirms that the parametric chi-squared test can produce biased results in small frequency domains with unevenly distributed samples, whereas the corresponding permutation test does not. This involves an artificial dataset that exhibits no actual association between class and attribute values. For each class, an equal number of instances with random attribute values is generated. The total number of instances is set to twenty. The attribute values are skewed so that more samples lie close to the zero point. This is done using the distribution $\lfloor kx^2 \rfloor$, where k is the number of attribute values and x is distributed uniformly between 0 and 1. The estimated p-value of the chi-squared permutation test $\hat{p}_{\chi}$ and the p-value of the parametric chi-squared test p_{χ} are calculated for this artificial, non-informative, attribute. This procedure is repeated 1000 times, each time with a different random sample of twenty instances. In each of the 1000 trials, 1000 random permutations are used to estimate the p-value of the permutation test.

Table 4.1 shows the average values obtained using this procedure. It can be seen that p_χ decreases systematically as the number of attribute values increases, whereas this is not the case for $\hat{p}_\chi$. Note that the variation in $\hat{p}_\chi$ is due to the sparseness of the test. This result shows that p_χ is too liberal a test for this particular dataset when there are many classes and attribute values. There also exist situations in which it is too conservative (Good, 1994). If used for pre-pruning, a test that is too liberal does not prune enough, and a test that is too conservative prunes too much.

As discussed in the previous chapter, enumerating all permutations for a permutation test is infeasible except for very small samples (Section 3.2.3). However, the p-value can be approximated to arbitrary precision using Monte Carlo sampling in conjunction with the sequential probability ratio test (Section 3.2.4). Another problem is the sparseness of the distribution of p-values for small samples, which can lead to overly conservative p-values. As mentioned in the previous chapter, statisticians recommend the mid-p value as a remedy for this problem (Section 3.2.6).

The χ^2 statistic is not the only test statistic that can be used in a permutation test. Another potentially useful candidate statistic for pre-pruning, which we also evaluate empirically, is the exact probability of a contingency table given its marginal totals. The corresponding test is known as the Freeman and Halton test, and has been used in conjunction with reduced-error pruning in the previous chapter (Section 3.2.5). It constitutes the extension of Fisher's exact test to multi-dimensional contingency tables.

4.1.2 Multiple tests

Pre-pruning involves significance tests at each node of the decision tree. At any particular node, each attribute that has not been used higher in the tree is subjected to a test at a fixed significance level α. The idea is that the test rejects a random attribute for which the null hypothesis is true in a fraction $1-\alpha$ of all cases. Given that the assumptions of the test are fulfilled, this is in fact the case for each *individual* attribute. However, the situation is different when the efficiency of the test with respect to all m attributes at the node is considered, because the probability of finding at least one significant attribute among m random attributes can be much higher than α (Jensen & Schmill, 1997). If the m attributes are independent, the significance level α_c of the combined test is given by the following equation:

$$\alpha_c = 1 - (1 - \alpha)^m. \tag{4.1}$$

To obtain a particular overall significance level, the significance level α for each individual test must be adjusted according to:

$$\alpha' = 1 - \sqrt[m]{(1 - \alpha_c)}. \tag{4.2}$$

Then, an attribute is considered to be insignificant if the test's p-value for this attribute is greater than this adjusted value α'. Equivalently, instead of adjusting the significance level α, the p-value p_i for a particular attribute i can be adjusted according to

$$p'_i = 1 - (1 - p_i)^m. \tag{4.3}$$

and compared to the original significance level α. This has exactly the same effect.

This adjustment is often called the "Bonferroni correction."[1] If the number of attributes at each node were constant, this correction would not be necessary: the significance level could be adjusted externally to maximize predictive accuracy. However, because the number of attributes that are tested may be different at each node of the tree, α should be adjusted locally to maximize the efficiency of pruning.

In practice, the Bonferroni correction is only a heuristic. It assumes that the p-values for the different attributes are independent random variables. This is rarely the case in real-world datasets because the attributes themselves are often correlated. With correlated p-values, the Bonferroni correction produces an overly conservative significance level, and overpruning might occur (Jensen, 1992). An extreme example is the case where all attributes tested at a particular node A are copies of each other. In that case, *no* adjustment of the p-value is required. However, it is possible that the Bonferroni correction is justified at another node B in the same tree because the attributes tested at that node exhibit little correlation with each other. This

[1] Note that the correct term for this adjustment is actually "Šidák correction" (Westfall & Young, 1993). The "real" Bonferroni correction is only an approximation, defined to be $p'_i = min(m \times p_i, 1)$ (Westfall & Young, 1993). We use the term "Bonferroni correction" here in order to be consistent with Jensen et al. (1997).

is a problem because global adjustment of the significance level is likely to either (a) incorrectly prevent further splitting at node A or (b) incorrectly allow further branching at node B.

Statisticians have devised a method that automatically accounts for correlated attributes: a permutation test on the *p-values* of the underlying significance test (Westfall & Young, 1993). Let p_i be the p-value of the underlying test for a particular attribute i. The permutation test computes the probability that, among all permutations of the data, the underlying test will output a p-value at least as small as p_i for any of the m attributes. In other words, the permutation test computes the probability that, among all m attributes, a p-value as low as p_i will be observed if the null hypothesis is correct. This probability constitutes the adjusted p-value p'_i for attribute i. When all attributes are independent of each other and the class, this permutation test generates the same adjusted p-value as the Bonferroni correction.

Unfortunately even a Monte Carlo approximation of the permutation procedure for multiple tests is computationally very expensive. It cannot be used when the underlying test is also a permutation test because this would involve a nested loop of Monte Carlo approximations—which is computationally prohibitive. However, even if the underlying test is parametric, the computational expense is considerable because the parametric test must be performed once for each random permutation of the data and each of the attributes involved until the adjusted p-value of all the attributes is approximated sufficiently well. Attributes that are clearly significant or insignificant cannot be excluded from the computation at an early stage, as in the case where only one attribute is involved in each test. In practice, these drawbacks outweigh the method's potential benefits. We performed initial experiments with this procedure in conjunction with the parametric chi-squared test and found that the resulting adjusted p-values were overly large if a dataset contained attributes with many values, leading to severe overpruning. The reason for this behavior is that the p-value of the parametric chi-squared test is optimistically biased for an attribute A with many values, thus making it hard to find *any* attribute with a p-value on the original data that is lower than A's p-value on most random permutations of this data. Because of this pathology, we restrict attention to the Bonferroni correction.

4.2 Experiments

Exact permutation tests, as well as adjustments for multiple tests, are promising candidates for an improved pre-pruning procedure. However, although theoretically justified, it remains to be seen whether they have any impact in practice. To clarify this point, this section presents experimental results for the 27 UCI datasets from Chapter 1, investigating the effect of each modification in detail. Prior to the experiments, all numeric attributes in the datasets are discretized into four intervals using equal-width discretization.

The first experiment compares the standard pre-pruning procedure based on the parametric chi-squared test to non-parametric methods based on permutation tests. The second experiment investigates whether results can be improved by adjusting for multiple tests. The final set of experiments compares pre-pruning with post-pruning, using the same statistical techniques for both methods.

4.2.1 Parametric vs. non-parametric tests

As discussed above, permutation tests, which are inherently non-parametric, have a theoretical advantage over parametric tests in small-frequency domains and in situations with very skewed class distributions. This section investigates whether this theoretical advantage is relevant in practice. We employ the pre-pruning procedure of Figure 4.1 to generate decision trees using both the parametric chi-squared test, the corresponding permutation test, and the Freeman and Halton test. When performing a permutation test, we use the sequential probability ratio test for the Monte Carlo approximation. Results for the mid-p value adjustment are also included in the comparison.

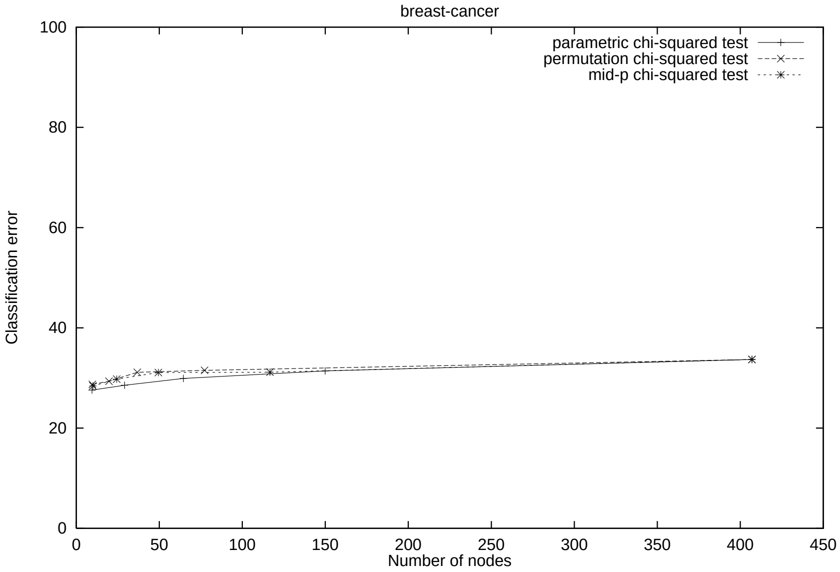
All experimental results are based on a plain ID3 decision tree learner that uses the information gain criterion to measure the strength of the association between an attribute and the class. In conjunction with the parametric chi-squared test, this is exactly the configuration used by Quinlan in the original version of ID3 (Quinlan, 1986b). Some of the datasets contain instances with missing attribute values. They are dealt with in the simplest possible way: at each node instances with missing values are assigned to the most popular branch.

To compare different pruning methods on a particular dataset, this chapter uses the same experimental methodology as the previous one: it applies each significance test at four different significance levels and estimates the accuracy and size of the resulting tree for each of them. The four significance levels are: 0.01, 0.05, 0.1, and 0.2. They are chosen to be conservative because the tests' p-values are known to be optimistic: they are not adjusted for multiple tests. The corresponding estimates are derived by averaging the results of ten-fold cross-validation repeated ten times.

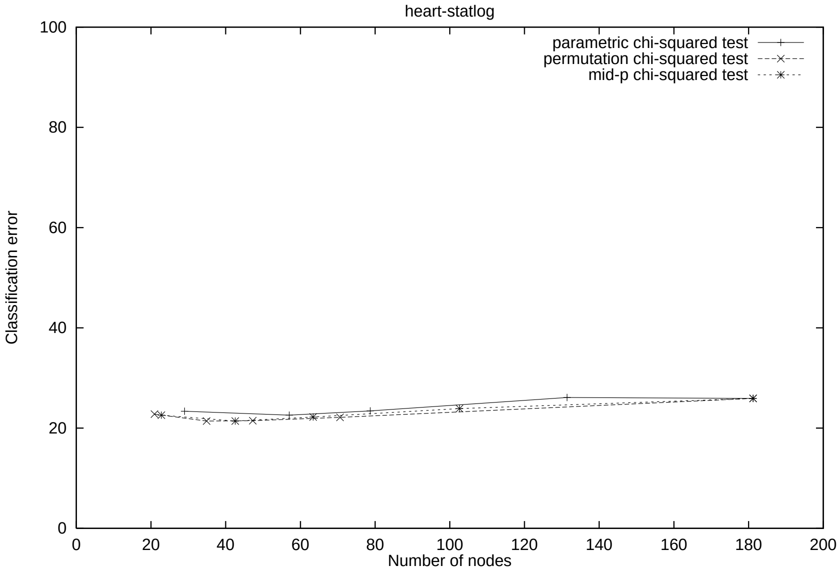
As discussed in the previous chapter on pages 50 and 51, pruning methods can only be compared in a fair way if both the accuracy *and* the size of the resulting trees are taken into account. Hence diagrams like the one in Figure 3.3 are required to draw conclusions about their relative performance. Figure 4.5 shows corresponding diagrams for three of the 27 datasets—the remaining ones can be found in Appendix B.1. Note that these diagrams do not include the curves for the Freeman and Halton test and its mid-p value version: they are indistinguishable from those for the tests based on the χ^2 statistic. The diagrams in Figure 4.5 have been chosen because they each exhibit a typical behavior for the five significance tests included in the comparison. The graphs also contain results for unpruned trees, which corresponds to applying the significance tests with a significance level of 1. This is the rightmost data point in each of the diagrams, and it is shared by all three of the curves representing the different significant tests.

The results show almost no difference between the two versions of the permutation test if the significance level is adjusted appropriately. The mid-p version of the test prunes less aggressively but this can be rectified by changing the significance level. This is the case for all three datasets displayed in Figure 4.5. However, for some datasets, the graphs for the parametric test are located either below or above the corresponding graphs for the permutation tests. One example where the permutation tests perform slightly better than the parametric test is the heart-statlog dataset of Figure 4.5b; another example is the sonar dataset of Figure B.1d. On the vowel dataset of Figure 4.5c, the permutation tests perform slightly better only if aggressive pruning is performed. On the other hand, on the breast-cancer dataset, shown in Figure 4.5a, the parametric test appears to have a slight edge—although the minima of the three curves are almost identical. Another example where the

(a)



(b)



(c)

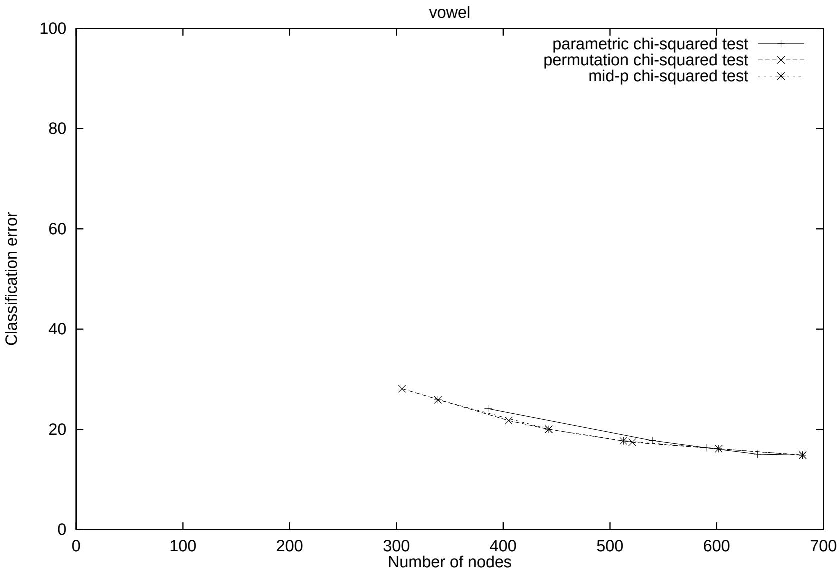


Figure 4.5: Comparison of significance tests

parametric test performs marginally better is the audiology dataset of Figure B.1e.

In the majority of cases there is virtually no difference in performance between the three tests if the significance levels are adjusted appropriately. Moreover, on those datasets where there is a difference, it is usually small. Thus it is questionable whether the additional computational expense for the permutation tests is worthwhile.

4.2.2 Adjusting for multiple tests

The failure to adjust for multiple significance tests is another potential drawback of the standard pre-pruning procedure. This section presents experimental results clarifying the importance of appropriate adjustments in this application. We compare the performance of two versions of pre-pruning based on the parametric chi-squared test. The first version does not adjust for multiple tests; the second uses the Bonferroni correction to do so.

Figure 4.6 shows three diagrams exemplifying the relative performance of these two versions of the pre-pruning algorithm on the benchmark datasets. The diagrams for the remaining 24 datasets are presented in Appendix B.2. As in the previous section, plain ID3 was used as the base inducer, and performance was estimated by repeating ten-fold cross-validation ten times. For the Bonferroni correction, the four global significance levels are set to 0.1, 0.2, 0.3, and 0.4 because the Bonferroni-adjusted pruning procedure is inherently more aggressive. Without the correction they are set to 0.01, 0.05, 0.1, and 0.2, as in the last section.

The Bonferroni correction is different from a *global* adjustment of the significance level because it adjusts the significance level *locally* at each node of the decision tree according to the number of attributes undergoing a significance test. Therefore one would expect the resulting decision trees to be different from the ones obtained by setting the significance level to a fixed value for all nodes of the tree. However, the experimental results for the benchmark datasets show that there is very little difference in performance between these two procedures if the global significance level is set to an appropriate value. For all datasets, combining the procedure's graphs produces a smooth result. The only major difference is that the ordinary test covers a wider range of tree sizes. These results imply that the Bonferroni correction has

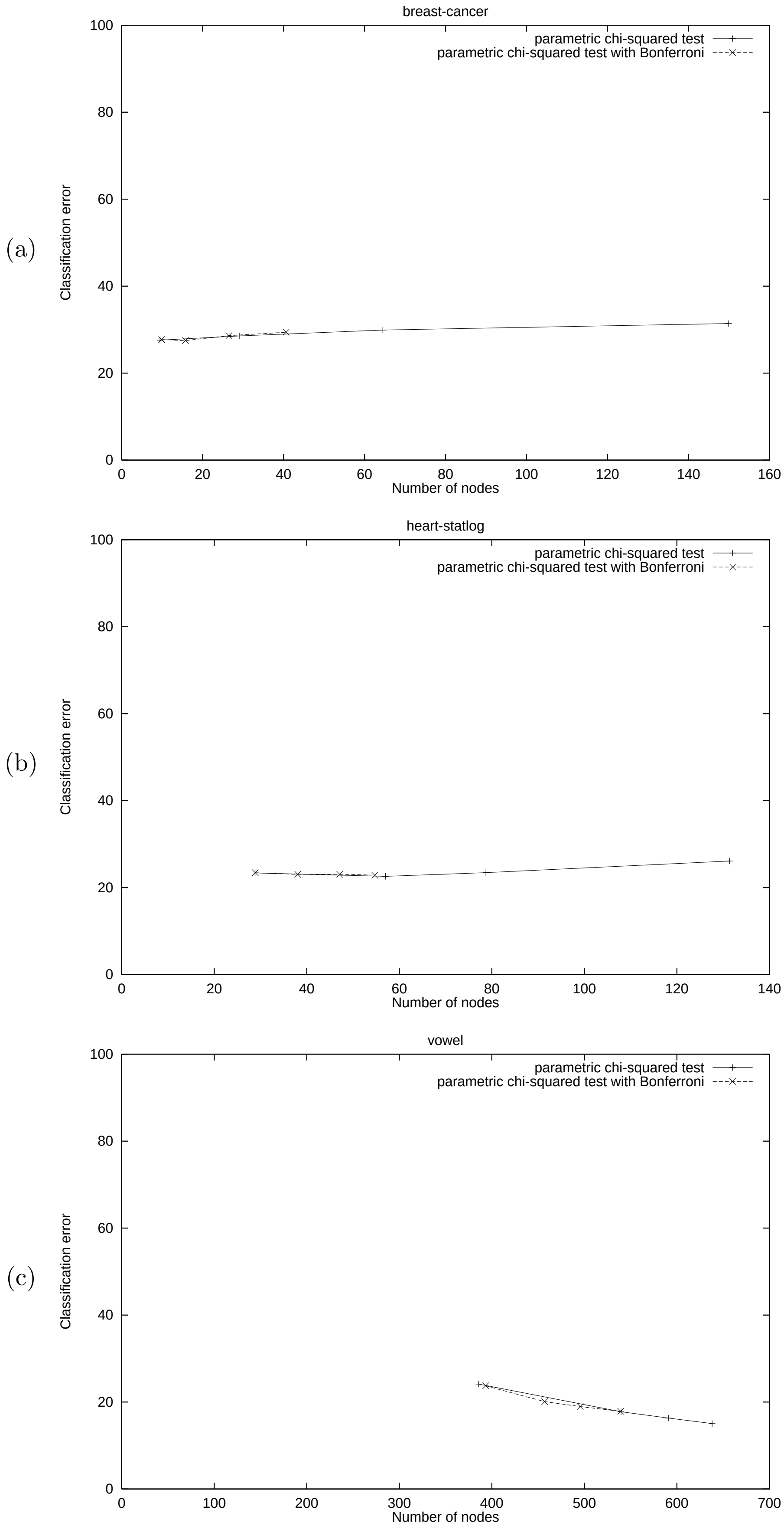


Figure 4.6: Using the Bonferroni correction

very similar values at each node of the decision tree; it does not produce widely different results in different parts of the tree.

4.2.3 Pre- vs. post-pruning

As discussed in the introduction to this chapter, pre-pruning is assumed to be inferior to post-pruning because of interaction effects between attributes that are only visible in the fully grown tree. However, empirical comparisons between pre- and post-pruning methods are rare. The few reports that do exist use different statistical techniques to implement the two pruning paradigms.

We present results of an experiment comparing pre- and post-pruning on a level footing. Both paradigms are implemented using the same basic statistical technique. For pre-pruning, the ordinary parametric chi-squared test is used for stopping. For post-pruning, the same test is used to filter irrelevant attributes at each node of the tree. However, in contrast to the pre-pruning procedure in Figure 4.1, splitting does not stop if no significant attribute can be found according to the given significance level: it continues with the attribute exhibiting the lowest p-value. Consequently this procedure can unearth splits with significant p-values even if they occur below a node with an insignificant association. When the tree has been fully expanded, pruning proceeds in the standard bottom-up fashion employed by most post-pruning algorithms: a subtree is considered for replacement by a leaf after all its branches have been considered for pruning. Pruning decisions are made according to the p-values observed during training. Replacement occurs whenever all the successors of a node are leaves and the node's p-value exceeds the significance level. This post-pruning procedure is very similar to critical value pruning (Mingers, 1987). The only difference is that critical value pruning always chooses the attribute with the lowest p-value for splitting; it does not distinguish between the strength and the significance of an association.

As mentioned earlier, the parity problem is an artificial learning task for which pre-pruning algorithms are expected to fail. This is experimentally verified in Figure 4.7, which shows the performance of the pre- and post-pruning algorithms for the parity problem with five attributes. More specifically, it shows the size of the pruned tree in number of nodes for several different training set sizes. A significance

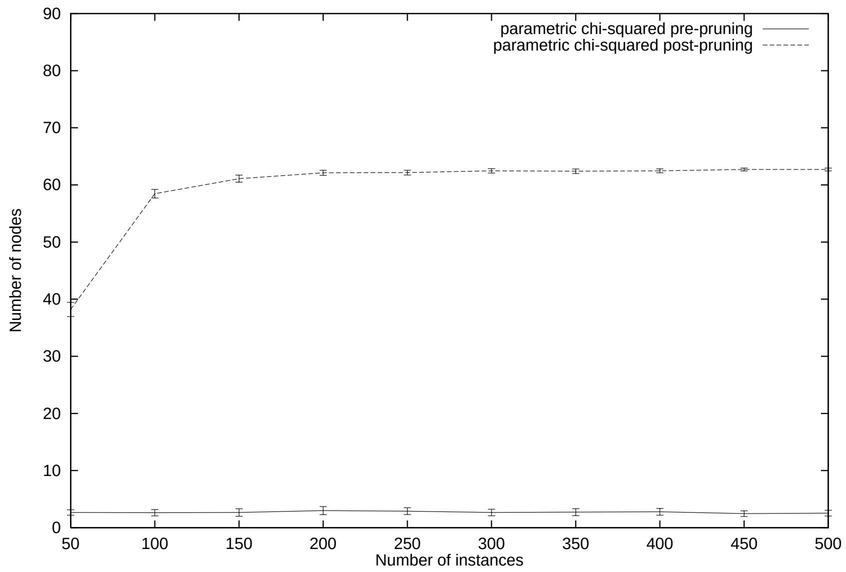


Figure 4.7: Pre- vs. post-pruning on the parity dataset

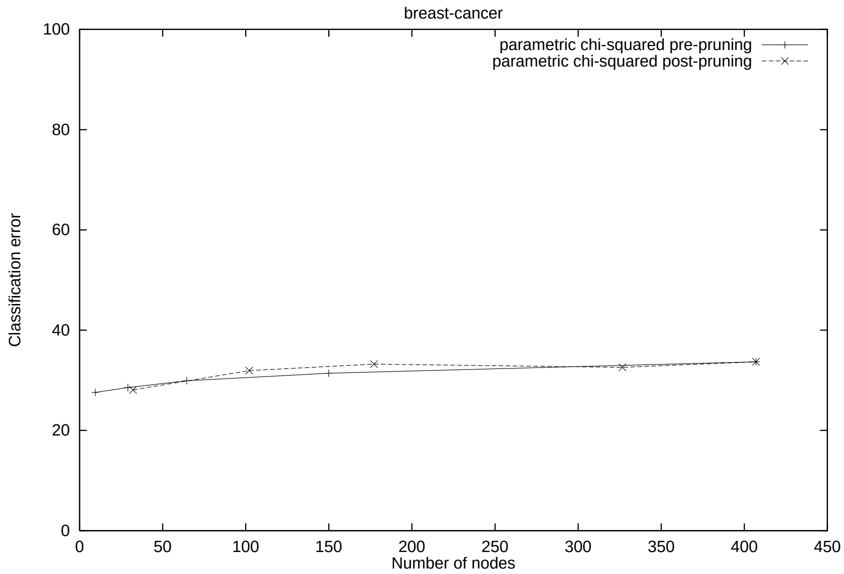
level of 0.1 is used for both pre- and post-pruning. Each data point is the result of repeating the same experiment with 100 different randomly generated dataset-s of the same size. As explained in Section 3.3, classification accuracy increases monotonically with tree size in this problem.

Figure 4.7 shows that post-pruning only needs a couple of hundred instances to closely approximate the correct tree for this problem. The performance of pre-pruning is independent of the number of training instances that are supplied: it fails to identify any structure in the dataset. These results show that the above implementations of the two pruning paradigms, although based on the same basic statistical technique, behave as expected for the parity problem. Consistent with Hypothesis 4.3 from the introduction, they suggest that post-pruning is the method of choice.

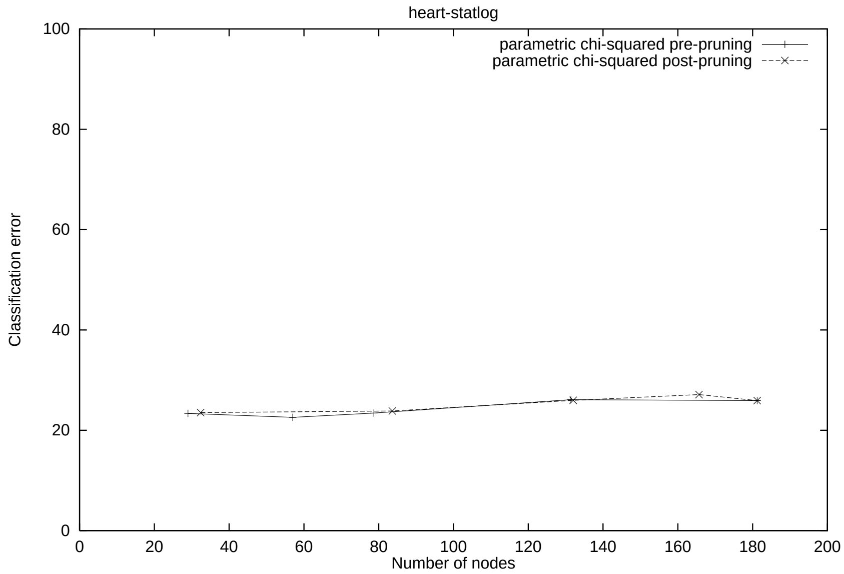
There remains the question how frequently parity-type situations arise in practical datasets. As explained in the introduction, it is likely that all relevant attributes in a dataset are predictive of the class in practice. If the predictive power of an attribute is strong enough to make the association statistically significant, pre-pruning can succeed.

Experiments on practical datasets are the only way of checking whether post-pruning has a competitive edge in real-world learning tasks. Figure 4.8 shows dia-

(a)



(b)



(c)

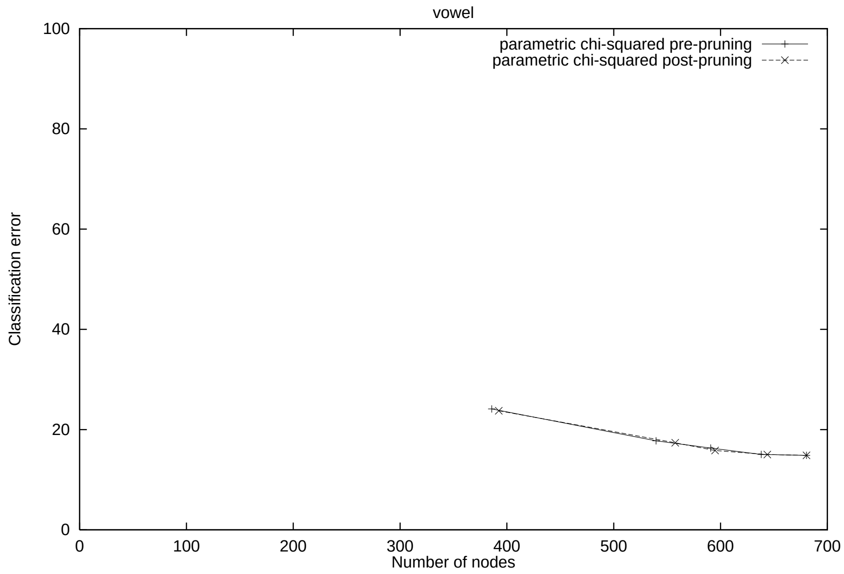


Figure 4.8: Pre- vs. post-pruning

grams for three of the 27 datasets from above, comparing the performance of pre-pruning and post-pruning using data points corresponding to four significance levels: 0.01, 0.05, 0.1, and 0.2. The graphs also contain a data point for the unpruned tree, which corresponds to a significance level of 1.0. The remaining 24 diagrams can be found in Appendix B.3. As above, estimates are derived by repeating ten-fold cross-validation ten times.

Surprisingly, none of the graphs show more than marginal performance differences between the two pruning paradigms. The graphs suggest that performance differences can be eliminated by adjusting the significance level for the two procedures individually. They show that the two paradigms only differ in the amount of pruning incurred for a fixed significance level: post-pruning is more liberal than pre-pruning and prunes less aggressively. There are two factors that are likely to be responsible for this difference in behavior. First, as discussed above, the parametric chi-squared test is often overly liberal for very small samples. These samples occur frequently at the leaves of the unpruned tree, which makes post-pruning automatically less aggressive than pre-pruning. Second, post-pruning performs many more significance tests than pre-pruning because it considers all the nodes in the fully grown tree for pruning, not just the ones expanded so far. Therefore, it is more likely to find significant associations *just by chance*. This also leads to additional structure. In some datasets, this extra structure is advantageous because the domain requires very little pruning. However, the experimental results show that the same effect can be achieved by increasing the significance level for pre-pruning. This also reduces the amount of pruning that is done, and the difference in performance vanishes.

These results demonstrate that, on practical datasets, interaction effects have very little influence on performance differences between pre- and post-pruning. For optimum performance, the only problem is to find the right significance level. This drawback is shared by both pre- and post-pruning. As in the last chapter, the best significance level can be found automatically using cross-validation.

The results do not show that pre-pruning based on the chi-squared test is competitive with *every* post-pruning method. However, they strongly suggest that potential differences in performance are not due to interaction effects but must be attributed

		attribute values: +	attribute values: −	
class values	A	42	67	109
	B	0	31	31
		42	98	140

Figure 4.9: A split that does not reduce the classification error

to other properties of the pruning procedure.

One possible candidate is the fact that most post-pruning methods are error-based: their pruning decisions are guided by a criterion that is based on the classification error (on either the training data or a separate pruning set). Using the error makes sense because the ultimate goal is to derive a decision tree that achieves minimum classification error on fresh data. Pre-pruning methods, on the other hand, cannot use an error-based criterion for stopping because it prevents potentially useful splits from being expanded. An example shows why. Figure 4.9 exhibits a contingency table for a hypothetical split that does not decrease the classification error although it is potentially useful because it produces a "pure" node. In conjunction with other splits further down the tree it might ultimately lead to a very accurate decision tree. The split does not decrease the classification error because the majority class is the same in both of its subsets. Thus, with respect to the classification error, it appears to be completely useless. This property of error-based criteria has first been observed in the closely related problem of global discretization (Kohavi & Sahami, 1996), where numeric attributes are discretized into intervals prior to induction.

The experimental results invite the question of when pre-pruning should be used instead of post-pruning in practical applications. Pre-pruning is faster than post-pruning because it does not have to generate a fully-grown decision tree: the experimental results show that it is not necessary to further decrease the significance level once the estimated classification error on fresh data starts to increase. The potential for speed-up is particularly pronounced for large datasets with little structure that require aggressive pruning. However, post-pruning performs as well as pre-pruning on all benchmark datasets included in the comparison, and it is guaranteed to succeed in parity-type situations. Moreover, it can potentially benefit from error-based pruning criteria. Consequently, to be on the safe side, we recommend post-pruning methods in practice if training time is not a critical factor of the application at hand.

4.3 Related work

Historically, pre-pruning techniques were used long before post-pruning methods were investigated. The earliest decision tree inducers employed pre-pruning to deal with noisy datasets. Since then, pre-pruning was largely abandoned, due mainly to influential books by Breiman et al. (1984) and Quinlan (1992).

4.3.1 Pre-pruning in statistics and machine learning

In statistics, CHAID (Kass, 1980) was the first system for inducing classification trees.[2] CHAID employs the parametric chi-squared test for both split selection and pre-pruning. It can process nominal variables as well as ordinal ones. To generate a split on an attribute, CHAID uses the chi-squared test to merge attribute values greedily into subsets. Merging stops when all subsets are significantly different from one another according to the chi-squared test. The Bonferroni correction is used to adjust for multiple significance tests during this merging process. When an attribute is selected for splitting, a branch is extended for each of the subsets. Splitting stops when no significant split can be found.

In contrast to the ID3 decision tree inducer used in this chapter, CHAID does not split on all values of a nominal attribute because it can merge attribute values. Thus it avoids unnecessary fragmentation of the training data, which is likely to produce more accurate trees. To keep things as simple as possible, our investigation uses the multi-way split employed by ID3 because our focus is on comparing pre- and post-pruning experimentally.

CHAID does not adjust for multiple tests *among* attributes, it only adjusts for multiple tests when merging attribute values. Jensen et al. (1997) use the CHAID procedure in conjunction with critical value pruning (Mingers, 1987) to post-prune decision trees. They do not compare pre-pruning with post-pruning. In addition to standard CHAID significance testing, they use the Bonferroni correction to adjust

[2] Although one of the oldest inducers around, CHAID is still very popular. It is, for example, part of SPSS' data mining software. A web search for "CHAID" at the time of writing returned 1070 hits.

for multiple tests among the attributes tested at a particular node. However, they do not show that this modification improves performance. The experimental results from this chapter suggest that adjusting for multiple tests among attributes in this fashion is not worthwhile.

The FACT method, also developed by statisticians, uses linear discriminant analysis in conjunction with ANOVA tests for split selection and pre-pruning (Loh & Vanichsetakul, 1988). A later version, called QUEST (Loh & Shih, 1997), employs quadratic discriminant analysis and the parametric chi-squared test in conjunction with the Bonferroni correction. QUEST can be used for pre- and post-pruning. However, a comparison of the two paradigms using QUEST has not been reported.

In machine learning, Quinlan's ID3 program (1986b), whose development reaches back to 1979, was the first top-down induction algorithm for decision trees (Quinlan, 1979). Like CHAID it uses the chi-squared test to decide when to stop growing a branch (Quinlan, 1986a). However, it does so in a slightly different way. Whereas CHAID selects a split by minimizing the test's p-value, ID3 uses it to filter irrelevant attributes. From the remaining attributes, it chooses the one that maximizes the information gain. Thus ID3 distinguishes between the strength and the significance of an attribute's association to the class.

Fisher and Schlimmer (1988) compare the prediction accuracy of pre-pruned and unpruned ID3 trees. They find that the more statistically independent the class is of the predictor attributes, the more pruning improves performance. In other words, pruning is particularly beneficial when the attributes have little predictive power. They also find that pruning can be harmful when there are very few training instances. Fisher (1992) extends these results by varying the number of training instances, the amount of attribute and class noise, and the skewness of the class distribution. He also considers different confidence levels for pruning. Using data from three practical domains he shows that, for sparse training data and low noise levels, aggressive pruning can be detrimental to performance. However, pruning has little influence on accuracy if the class distribution is highly skewed.

In later work, Quinlan (1987a) replaced ID3's pre-pruning procedure with various post-pruning methods. In his book on C4.5 (Quinlan, 1992) he argues that pre-pruning is necessarily inferior because of the horizon effect mentioned in the introduction to this chapter. However, he does not present experimental results to support this argument.

In their CART system for building classification trees, Breiman et al. (1984) also abandoned pre-pruning in favor of a post-pruning approach: they do not provide empirical results supporting their decision either. Their approach to pre-pruning is based on the value of the splitting criterion—for example, the information gain. They propose to stop splitting when the splitting criterion fails to exceed a certain threshold. This is known to be problematic because the information gain, as well as other impurity-based splitting criteria—for example, the Gini index (Breiman et al., 1984)—are not unbiased: their value depends on the number of attribute and class values (Kononenko, 1995). A similar approach to pre-pruning has also been suggested in other early work on decision tree inducers (Sethi & Sarvarayudu, 1982; Goodman & Smyth, 1988). Cestnik et al. (1987) use thresholds for the splitting criterion, the probability of the majority class, and the relative node weight, in a combined stopping criterion. With this pre-pruning procedure they obtain more accurate trees than minimum-error pruning (Niblett & Bratko, 1987) on one of four datasets investigated. Hong et al. (1996) present a way of normalizing a splitting criterion by computing its expected value under the null hypothesis. However, there is no theoretical justification for using this normalization procedure instead of a significance test.

4.3.2 Pre-pruning with permutation tests

Several authors have suggested using a permutation test instead of the parametric chi-squared test in pre-pruning decision trees. Niblett (1987) argues that the one-sided version of Fisher's exact test is preferable to the parametric test because of the small samples occuring in a decision tree; however, he does not report experimental results. The one-sided version of Fisher's exact test is restricted to 2×2 contingency tables, which means that it can only be used for binary trees and two-class problems. Li and Dubes (1986) also use Fisher's exact test for pre-pruning and show that it works better than an entropy-based stopping criterion. This chapter presents experimental results for the Freeman and Halton test, which is the extension of Fisher's exact test to multi-class settings.

Table 4.2: Average probabilities for random data (600 instances; uniformly distributed attribute values)

Attribute Values	Class Values	(a) $\hat{p}$	(b) p_f	(c) p_χ
2	2	0.525	0.045	0.488
2	5	0.511	1.63e-05	0.509
2	10	0.506	1.80e-10	0.505
5	2	0.497	1.55e-05	0.496
5	5	0.500	9.25e-18	0.497
5	10	0.491	6.62e-35	0.487
10	2	0.498	1.77e-10	0.495
10	5	0.520	7.84e-35	0.515
10	10	0.512	4.89e-68	0.503

Martin (1997) uses the exact probability of a contingency table given its row and column totals for pre-pruning. This probability is given by the multiple hypergeometric distribution. It is the *test statistic* of the Freeman and Halton test. Like other test statistics—for example, the chi-squared statistic—its value depends on the number of rows and columns in a contingency table. When used for pre-pruning, its value depends on the number of attribute values and the number of classes in the dataset.

It is straightforward to show, by adopting the experimental setup of White and Liu (1994), that the exact probability p_f is biased. As above, this involves an artificial dataset that exhibits no actual association between class and attribute values. For each class, an equal number of instances with random, uniformly distributed attribute values is generated. The estimated p-value of the Freeman and Halton test $\hat{p}$, the exact probability p_f, and the p-value of the parametric chi-squared test p_χ are calculated for this artificial, non-informative, attribute. This procedure is repeated 1000 times, each time with a different random sample of 600 instances. In each of the 1000 trials, 1000 random permutations are used to estimate the p-value of the Freeman and Halton test.

Table 4.2 shows the average values obtained. It can be seen in column (b) that p_f systematically decreases with increasing number of classes and attribute values. Even more importantly, it is always close to zero. If used for pre-pruning at the 0.01 level (as proposed by Martin, 1997), it would fail to stop splitting in every situation except that represented by the first row. Hence it is inadvisable to use

p_f for pre-pruning. On the other hand, neither $\hat{p}$ nor p_χ varies systematically with the number of attribute and class values. As before, the variation in $\hat{p}$ is due to the sparseness of the test.

Martin (1997) reports that pre-pruning using the exact probability is superior to pre-pruning with the parametric chi-squared test. Given the results presented in this chapter, this finding is questionable. A careful look into the experimental set-up he used reveals the reason for the discrepancy. Martin employs Cochran's (1954) criterion to determine when the mathematical assumptions for the parametric chi-squared test are fulfilled. If Cochran's criterion is not met, an attribute is not filtered even if its p-value is lower than the significance level. Unfortunately, Cochran's criterion is *almost never* met after the first few splits in a decision tree. This means that pre-pruning *almost never* stops splitting. The resulting trees are rarely different from an unpruned tree—Martin acknowledges this fact. Thus it is hardly a surprise that pre-pruning with the chi-squared test does not perform well in Martin's experiments.

Martin (1997) also reports that pre-pruning using the exact probability is superior to post-pruning with C4.5's pruning procedure (Quinlan, 1992). Curiously, in Martin's experiments, post-pruned trees are never more accurate than unpruned ones and only slightly smaller. Martin (1997, page 276) notes that "It was somewhat surprising that C4.5's pessimistic pruning method was not more effective." We ran C4.5 (Quinlan, 1992) both with and without pruning on the discretized datasets from this chapter, with subtree raising turned off and default parameter settings, measuring performance using ten-fold cross-validation repeated ten times. According to a paired two-sided t-test at the 5% level, the pruned trees are significantly more accurate than the unpruned ones on 17 datasets, and less accurate on only three. Also, the pruned trees contained 51% fewer nodes, averaged across all datasets. In other words, C4.5's pessimistic post-pruning substantially reduces the size of the unpruned trees. This is consistent with results reported by other authors—for example, Mingers (1989). These observations cast doubt on Martin's finding that post-pruning is ineffective.

4.3.3 Statistical tests for attribute selection

Statistical tests have also been used for the purpose of attribute selection only. White and Liu (1994) compare several entropy-based selection criteria to parametric tests that rely on the chi-squared distribution. More specifically, they compare the entropy-based measures to parametric tests based on both the chi-squared and log likelihood ratio statistics. They conclude that each of the entropy measures favors attributes with larger numbers of values, whereas the statistical tests do not suffer from this problem. However, they also mention the problem of small expected frequencies with parametric tests, and suggest the use of Fisher's exact test as a remedy.

Kononenko (1995) repeated and extended these experiments, and investigated several other attribute selection criteria as well. He shows that a chi-squared test based on the log likelihood ratio is biased towards attributes with many values if the number of classes and attribute values relative to the number of instances exceed the corresponding figures considered by White and Liu (1994). This is not surprising: it can be traced to the problem of small expected frequencies. For the log likelihood ratio the effect is more pronounced than for the chi-squared statistic (Agresti, 1990).

Kononenko also observes another problem with statistical tests. The restricted floating-point precision of most computer arithmetic makes it difficult to use such tests to discriminate between different *informative* attributes. The reason for this is that the association to the class is necessarily highly significant for all informative attributes.[3] However, there is an obvious solution, which is used in our experiments: once it has been established that an attribute is significant, it can be compared to other significant attributes using an attribute selection criterion that measures the strength of the association—for example, the information gain.

4.4 Conclusions

This chapter investigates ways of improving the standard procedure for pre-pruning decision trees based on the parametric chi-squared test and compares the performance of pre- and post-pruning on a level footing.

[3]The probability that the null hypothesis of no association between attribute and class values is incorrectly rejected is very close to zero.

The first hypothesis is that permutation tests based on the chi-squared statistic improve on the parametric chi-squared test because they are statistically valid for the small samples occurring in a decision tree. However, the experimental results show that there is very little difference between the permutation tests and the parametric test in this application if the significance levels are chosen appropriately. On some datasets the permutation tests appear to have a slight edge, while on others the parametric test does, but the differences are always very small. Another finding is that this result does not change when the chi-squared statistic in the permutation tests is replaced by the multiple hypergeometric probability.

The second hypothesis is that pre-pruning can be improved by adjusting for multiple significance tests among attributes. Experimental results obtained with the Bonferroni correction show that this is not the case. The effect of this correction, which adjusts the significance level locally at each node of the decision tree depending on the number of attributes that are considered for splitting, can be closely approximated in practice by adjusting the test's significance level globally.

The third hypothesis is that post-pruning outperforms pre-pruning because it has access to information about attribute interactions in the data. To allow for a fair comparison experiments are performed that use the parametric chi-squared test for both pre- and post-pruning. Surprisingly, the results show there is almost no difference in performance between the two paradigms on practical datasets. For a given significance level, post-pruning produces larger trees than pre-pruning. Sometimes these trees are more accurate, sometimes less. The difference can be largely eliminated by adjusting the significance level. Thus it appears questionable to motivate the use of post-pruning techniques using attribute interactions. Problems with the horizon effect seem to be restricted to artificial datasets. In practice pre-pruning might be a viable alternative to post-pruning because it is computationally more efficient.

This chapter concentrates on pruning mechanisms. To keep things simple it does not deal with issues such as numeric attributes or subsets of attribute values. If handled in an appropriate way, there is no reason to assume that these issues will influence the results presented here. Note, however, that mechanisms like the Bonferroni correction might be useful for finding splits on numeric attributes or for grouping nominal values into subsets (Kass, 1980). This chapter only shows that it is ineffective when used to adjust for multiple tests among attributes.

Chapter 5

Pruning decision lists

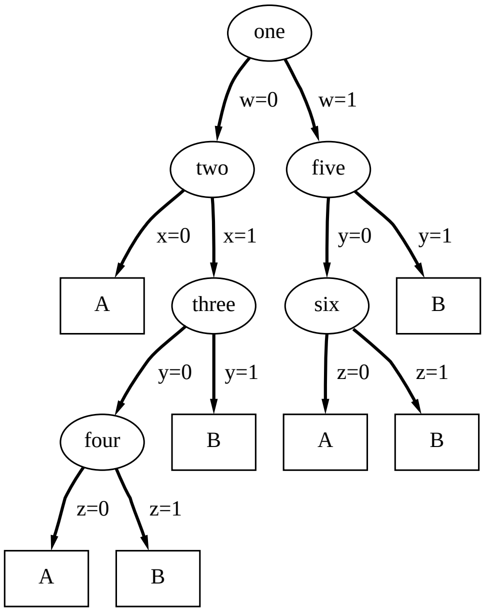
Decision trees are accurate classifiers that are easy to understand. However, in some domains their comprehensibility suffers from a problem known as subtree replication (Pagallo & Haussler, 1990). When subtree replication occurs, identical subtrees can be found at several different places in the same tree structure. Figure 5.1 shows an example domain where subtree replication is inevitable. The target concept, described by the simple rules in Figure 5.1a, leads to the rather complex decision tree in Figure 5.1b because information must be replicated in the subtrees attached to nodes three and five. The complexity of the decision tree cannot be reduced by changing the order in which the attributes are used for splitting: every correct decision tree for this problem has the same size.

Figure 5.1 illustrates that subtree replication occurs whenever an accurate description of the target concept consists of a set of rules, the rules include tests on disjoint sets of attributes, and they cannot be decoupled by different tests on the same attribute. Unfortunately there are many situations where this scenario is likely to hold in practice. For example, consider a disease that can be diagnosed using either one of two groups of symptoms, and the two groups are not mutually exclusive. In this application a decision tree is not a perspicuous representation of the underlying concept.

Fortunately there is an alternative to decision trees that avoids the problem of subtree replication: the target concept can be modeled as a set of rules, where each rule is a conjunction of attribute-value tests together with a corresponding class assignment. In other words, the goal is to induce the rules in Figure 5.1a directly from a set of data. This chapter presents simple yet surprisingly efficient algorithms for inducing rule sets. The main contribution is a careful examination of the pruning

if w=0 and x=0 then A
if y=0 and z=0 then A
otherwise B

(a)



(b)

Figure 5.1: An example of subtree replication

operations performed by a fast pruning algorithm known as "incremental reduced-error pruning" (Fürnkranz & Widmer, 1994; Fürnkranz, 1997), which leads to a combination of the two main paradigms for learning rule sets: obtaining rules from decision trees (Quinlan, 1987b) and separate-and-conquer rule learning (Fürnkranz, 1999).

The particular example in Figure 5.1 is a two-class learning problem. In this case it is sufficient to learn a set of rules describing only one of the classes, for example, class A. A test instance that does not satisfy any of the conjunctions for class A is assigned to class B. This means that there is only one rule for class B, called the "default rule." This default rule "fires" when none of the rules for class A fires. The assumption underlying the default rule is often called the "closed-world assumption" (Witten & Frank, 2000) because it presumes that an instance not belonging to class A belongs to class B and that there are no other classes. In practice, however, datasets have often more than two classes and the closed-world assumption cannot be applied.

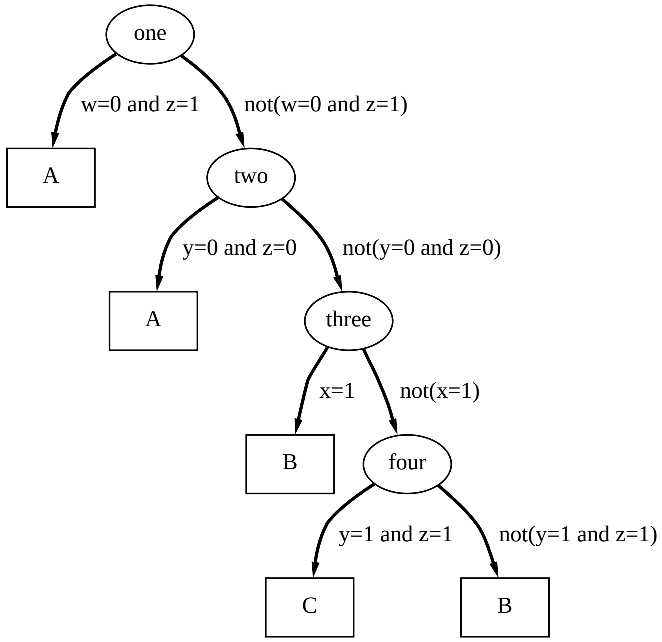
There are two ways of learning rules in the general multi-class setting. The first approach generates a rule set separately for each class (Cendrowska, 1987). This has the disadvantage that rule sets for different classes can "overlap" in instance

if x=0 and z=1 then A
if y=0 and z=0 then C
if x=1 then B
if y=1 and z=1 then A
otherwise B

(a)

if x=0 and z=1 then A
else if y=0 and z=0 then A
else if x=1 then B
else if y=1 and z=1 then C
else B

(b)



(c)

Figure 5.2: A simple decision list

space. Consequently a test instance can be assigned to more than one class, and some mechanism for arbitration must be applied to determine a prediction. Usually this is done by weighting the rules, for example by the number of training instances that they cover (Cendrowska, 1987). A test instance is assigned to the rule with the highest weight among all rules that fire. Unfortunately the weighting mechanism makes it difficult to interpret the rules: the weights must be taken into account to obtain insight into the knowledge represented by the classifier, and it is difficult for the user to judge which rules overlap and to what extent.

The second way of dealing with multi-class domains does not share this disadvantage. It prevents potential ambiguities by imposing an ordering on the rules. During classification the rules are evaluated sequentially and a test instance is classified using the first rule that fires. This procedure requires that the last rule in the list has a functionality similar to the default rule in the two-class case from above: it must fire for all instances that are not covered by one of the preceding rules—otherwise some instances would be left unclassified. Ordering the rules in this way assures that one and only one rule applies to any given test instance. It provides the user with a straightforward way of interpreting the classifier and makes all acquired knowledge explicit.

An ordered set of rules is called a "decision list" (Rivest, 1987). Figure 5.2a displays a decision list for a problem with three classes. The ordering information in this list can be made explicit by writing the rules using "else" statements, as shown in Figure 5.2b. The classifier can also be represented as a graph, displayed in Figure 5.2c. This graph closely resembles the graph for a decision tree, and in fact, a decision list can be interpreted as a particular type of binary tree (Pagallo & Haussler, 1990) in which the simple attribute-value tests that are performed at each node of an univariate tree are replaced by more complicated conjunctions of attribute-value tests, and at least one successor of each node is a leaf. Subtree replication is not possible in these extended decision trees.

This chapter introduces a new algorithm for building and pruning decision lists that applies the standard separate-and-conquer algorithm for rule learning in conjunction with an inducer for univariate decision trees. It produces a rule set by iteratively generating a series of pruned decision trees, reading one rule off each of them. An important feature of this algorithm is that the decision tree inducer differs from a standard decision tree learner in the way that pruning proceeds. Its pruning operations are specifically tailored to the objective of obtaining accurate rules. The resulting algorithm is a simple extension of the incremental reduced-error pruning algorithm for learning rule sets (Fürnkranz & Widmer, 1994), described in Section 5.1.2, that capitalizes on this improved pruning mechanism. In the following we call this improved procedure "tree-based incremental reduced-error pruning," or TB-IREP for short. The first hypothesis of this chapter is as follows:

Hypothesis 5.1. *TB-IREP produces more accurate rule sets than IREP.*

Chapter 3 shows that reduced-error pruning for decision trees can be improved by incorporating significance tests into the pruning procedure. It is likely that this finding also applies to TB-IREP. Corresponding to the empirical results from Chapter 3 it can be expected that significance testing reduces the size of the rule sets that are induced, and sometimes increases their accuracy. Since comprehensibility is the main motivation for learning rule sets, it is desirable to minimize the number of rules that must be considered by the user. This leads to a second hypothesis:

Hypothesis 5.2. *Significance testing makes the rule sets generated by TB-IREP* *both smaller and more accurate.*

TB-IREP builds a full decision tree in order to obtain a single rule. This rule is appended to the rule set, and the rest of the decision tree is discarded. However, it seems wasteful to grow and prune a full tree in order to obtain a single rule. And indeed, it turns out that this can be avoided by adopting a more conservative pruning strategy. The resulting procedure builds a "partial" decision tree instead of a full one. In a partial tree branches are left unexplored if they are not relevant for obtaining an accurate rule. This is achieved by integrating the growing and pruning operations of the decision tree inducer used by TB-IREP. It leads to a significant speed-up and cuts down the time that is needed for inducing the final rule set. In the following we call this algorithm PART. Although PART is faster than TB-IREP, it is possible that its conservative pruning strategy has a significant impact on the size and accuracy of the final rule set. To clarify this issue we test empirically the following hypothesis:

Hypothesis 5.3. *PART generates rule sets that are as accurate and as small as those produced by TB-IREP.*

The chapter is organized as follows. Section 5.1 first discusses global pruning of rule sets. Then we turn to incremental reduced-error pruning and how its pruning operations can be improved. This leads to the development of tree-based incremental reduced-error pruning. Section 5.2 explains how significance tests can be used to reduce the number of rules that this algorithm generates, and Section 5.3 describes the fast version based on partial trees. To test the above hypotheses, the performance of the different rule learning algorithms and the effectiveness of the proposed improvements is evaluated empirically in Section 5.4. Prior work on learning rule sets is reviewed in Section 5.5. The main findings of the chapter are summarized in Section 5.6.

5.1 Pruning algorithms

All separate-and-conquer rule learning schemes are modifications of the same simple algorithm, displayed in its most basic form in Figure 5.3. It starts with an empty set of rules and the full set of training instances. Then it attempts to find the single most

Procedure SeparateAndConquer(D: instances)

R := empty set of rules
while not D is empty
  r := best single rule for D
  R := add r to R
  remove those instances from D that are covered by r
return R

Figure 5.3: Separate-and-conquer algorithm

predictive rule for this dataset—where a rule consists of a conjunction of attribute-value tests and a class assignment—and adds it to the rule set. This rule covers a certain fraction of the training instances, namely all those for which the rule's tests are fulfilled. Consequently these instances are removed and the next rule is learned from the remaining ones. This process continues until all the training instances have been removed. Then the procedure halts and outputs the list of rules that have been generated. Because the rules are generated sequentially on increasingly smaller subsets of data, this algorithm automatically produces a decision list, that is, the rules are to be interpreted in the order in which they are generated.

5.1.1 Reduced-error pruning for rules

Rule inducers must address the problem of overfitting to obtain rule sets that work well in practice. In its simplest form, the algorithm from Figure 5.3 does not take any precautions against overfitting: a rule is refined by successively adding attribute-value tests until it only covers instances of one class. Then the rule is immediately put to use. However, the more conditions that are included in a rule, the fewer instances it will cover, and the more rules are needed to account for all the training instances. This reduces the classifier's comprehensibility and sometimes its predictive accuracy. Thus pruning strategies are an essential ingredient of any rule inducer that is useful in practice.

Global pruning

The earliest approaches to pruning rule sets are based on "global optimization." These approaches build a full, unpruned rule set using a strategy similar to the one outlined in Figure 5.3. Then they simplify these rules by deleting conditions from some of them, or by discarding entire rules. The simplification procedure is guided by a pruning criterion that the algorithm seeks to optimize. A popular criterion is the classification error of the rule set on a portion of the training data that has been put aside before the rules were generated. The pruning procedure seeks to minimize the error on this hold-out data. The rationale is that the error on this independent set of instances is an accurate estimator of the rules' performance on future data. This strategy bears strong similarities to reduced-error pruning for decision trees, discussed in Chapter 3, because it is based on splitting the original training data into two subsets—the first being used to generate the unpruned classifier and the second to prune it. It is no surprise that this pruning strategy for rule sets is also known as "reduced-error pruning" (Pagallo & Haussler, 1990; Brunk & Pazzani, 1991).

Problems

However, there is an important difference between reduced-error pruning for decision trees and rule sets: the subtrees of a decision tree do not overlap and can be pruned independently. Pruning decisions in one part of the tree do not influence pruning decisions in other parts. Thus reduced-error pruning for decision trees can be implemented as an elegant and fast bottom-up procedure that visits each node of the tree only once. Unfortunately the rules generated by separate-and-conquer learning are *not* independent (Fürnkranz, 1997). Deleting conditions from a rule—and thereby increasing its coverage—reduces the number of instances that must be classified by subsequent rules and also changes the class distribution in these instances. Thus pruning decisions cannot be made independently: pruning must be done globally. Each potential simplification of a rule must be evaluated with respect to the classification error of *all* subsequent rules, and every pruning decision depends on the outcome of all previous pruning decisions. This is why reduced-error pruning for rule sets must perform global optimization. The optimum pruning can only be found via exhaustive search. In practice, exhaustive search is infeasible and heuristic approximations must be applied—for example, by greedily deleting rules and conditions until the error on the hold-out data reaches a local minimum.

Even these approximate algorithms are quite time consuming, and their computational complexity depends critically on the size of the initial unpruned rule set.

However, computational complexity is not the only shortcoming of this global approach to simplifying rules (Fürnkranz, 1997). The problem of dependence between individual rules goes deeper: its effect reaches beyond the pruning process. Recall that the selection of attribute-value tests for a new rule depends on the instances that remain after previous iterations of the algorithm. Now consider what happens when one of the rules preceding the new rule is pruned: fewer instances reach the new rule and their distribution changes. However, the original set of instances was the one on which the rule was grown. The rule's attribute-value tests were chosen to optimize performance on this original set and might not be appropriate for the new, reduced, set. This problem cannot be rectified by pruning the new rule because pruning can only delete attribute-value tests—it does not allow for new, more suitable tests to replace the original ones. This is a distinct disadvantage of the global approach to pruning rules and a direct result of the dependence between individual rules. The next section discusses incremental reduced-error pruning (Fürnkranz & Widmer, 1994), an approach that avoids this problem of global optimization by interleaving the pruning and growing phases in a rule learning algorithm.

5.1.2 Incremental reduced-error pruning

Instead of growing a fully expanded rule set and pruning it in a separate step, there is a much simpler and faster approach that directly addresses the problem of dependence between individual rules and eliminates some drawbacks of global reduced-error pruning. The key idea is to prune a rule immediately after it has been built, before any new rules are generated in subsequent steps of the separate-and-conquer algorithm. In other words, before the algorithm proceeds, it removes all the instances that are covered by the *pruned* rule. This means that subsequent rules are automatically adjusted for changes in the distribution of instances that are due to pruning. By integrating pruning into each step of the separate-and-conquer algorithm, this procedure elegantly avoids the problem of dependencies that arises in the global approach.

Procedure IREP(D: two-class training data)

R := empty set of rules
while not D empty
  split D into growing set and pruning set
  r_u := best single rule for the positive class in the growing set
  r := pruning of r_u with the best performance on the pruning set
  if performance of r is not sufficient **then**
    r := empty rule assigning the negative class
  R := add r to R
  remove the instances from D that are covered by r
return R

Figure 5.4: Incremental reduced-error pruning

Incremental pruning

Figure 5.4 summarizes this algorithm, known as incremental reduced-error pruning or IREP (Fürnkranz & Widmer, 1994). By definition, the IREP algorithm can only be applied to two-class problems because it makes the closed-world assumption. Its heuristics are designed to find all pruned rules describing one of the two given classes—usually chosen to be the smaller one. In the following we call this class the "positive" class. The remaining parts of the instance space are assumed to be classified correctly using a default rule that assigns the second class—the "negative" class. Multi-class situations must be tackled by decomposing the learning task into several two-class problems; we will discuss later how this is done. In the following we will discuss IREP as implemented by Cohen (1995).

Like the standard separate-and-conquer algorithm, IREP starts with an empty set of rules, and constructs rules until the training data is exhausted. However, in each iteration it builds a *pruned* rule and removes the training instances that it covers. Like global reduced-error pruning it employs the hold-out method: two thirds of the data, called the "growing set," are used to generate a rule, and the rest, called the "pruning set," for pruning it. The data is stratified before it is split so that the class distribution is approximately the same in both sets. A rule is grown by maximizing the information gain for each attribute-value test that is added (Cohen, 1995). Because the algorithm focuses on rules for the positive class, the information gain is computed with respect to this class. If p_b (n_b) is the number of positive (negative) instances covered by the rule before the attribute-value test is added, and p_a (n_a) the analogous number after adding the test, the information gain can be written as

$$gain = p_a \left(-\log(\frac{p_b}{p_b + n_b}) + \log(\frac{p_a}{p_a + n_a}) \right).$$

When implementing this formula, it makes sense to initialize the positive and negative counts for the two classes to one so that the algorithm is less likely to create rules with very small coverage.

A rule is extended until no further improvements can be made, and pruned immediately afterwards. For pruning, the algorithm considers deleting the rule's last k attribute-value tests. For every value of k between one and the total number of attribute-value tests in the rule, it considers removing the last k attribute-value tests that have been added. It evaluates these potential simplifications on the pruning data, and chooses the one that maximizes the concentration of positive instances covered by the rule. In other words, it maximizes

$$concentration = \frac{p}{p+n},$$

where p is the number of positive instances in the pruning data covered by the rule, and n the corresponding number of negative instances.[1]

The concentration measure is an estimate of the rule's accuracy on fresh data. Consequently, if the accuracy of the best pruned rule is not sufficiently high, the algorithm discards the rule entirely, appends the default rule to the set, and terminates. In the implementation used by Cohen (1995), this happens when the rule's accuracy is lower than 0.5 because in that case letting the rule assign the negative class increases accuracy.

Time complexity

The IREP algorithm is very fast: each rule can be grown and pruned very quickly and the final rule set is small. Fürnkranz and Widmer (1994) state that the cost

[1]Note that Cohen (1995) uses $(p-n)/(p+n)$. This has exactly the same effect because

$$\frac{p-n}{p+n} = \frac{2p-(p+n)}{p+n} = 2\frac{p}{p+n} - 1.$$

for growing a rule is $O(n \log n)$, where n is the number of examples in the training set. However, this is only the case if the algorithm is not implemented efficiently and all n instances are scanned every time a new attribute-value test is selected. Assuming that numeric attribute values have been sorted in a pre-processing step, and that, after every attribute-value test, at most $(100/b)\%$ of the instances that passed the previous test remain to be processed, where b is bounded away from one, it is easy to show that the cost for growing a single rule is only $O(n)$. To see this, notice that there are at most $\log_b n_{grow}$ conditions in a rule, where n_{grow} is the number of examples in the growing set, and that the algorithm must go through n_{grow}/b^{i-1} instances to find the ith attribute-value test. This means that the overall time complexity for growing a rule is

$$\begin{aligned}
\sum_{i=0}^{\log_b(n_{grow})-1} \frac{n_{grow}}{b^i} &= n_{grow}\frac{\left(\frac{1}{b}\right)^{\log_b(n_{grow})-1+1} - 1}{\frac{1}{b} - 1} \\
&= n_{grow}\frac{1 - \frac{1}{n_{grow}}}{1 - \frac{1}{b}} \\
&= \frac{n_{grow} - 1}{1 - \frac{1}{b}} \\
&= O(n_{grow}) \\
&= O(n).
\end{aligned}$$

This result shows that a rule can be grown in time linear in the number of instances in the growing set.

The overall time complexity for inducing a rule also depends on the pruning step. Fürnkranz and Widmer (1994) find that the cost for pruning a rule is $O(n \log^2 n)$. However, this is too pessimistic a bound if only the tails of a rule are considered for pruning. Recall that a rule is pruned by cutting off the last k attribute-value tests. To find the best cut-point we must consider each of its tests once, starting from the end of the rule. This involves $O(\log n_{grow})$ evaluations. The distribution of pruning instances for each of the evaluations can be pre-computed by filtering all these instances through the fully grown rule. According to the derivation above this requires time $O(n_{prune})$. Hence the overall time-complexity for pruning a rule is

$$O(n_{prune}) + O(\log n_{grow}) = O(n),$$

where n is the total number of training instances.

Taken together, these findings mean that the cost for growing and pruning a single rule is linear in the number of instances if numeric attribute values have been pre-sorted. Assuming that the size of the final rule set is constant and does not depend on the size of the training set (Cohen, 1995), this implies that a complete rule set can be generated in time linear in the number of training instances. In practice, the number of rules is not constant and grows (slowly) with the size of the training data. Therefore the algorithm scales slightly worse than linearly on real-world datasets. The theoretical worst-case time complexity is $O(n^2)$, and it occurs when the number of generated rules scales linearly with the number of training instances.

Problems

Although this incremental pruning procedure avoids some of the problems related to global optimization, it does not solve the problem of rule dependencies completely. By pruning a rule before any further rules are generated, IREP assures that all subsequent rules are adapted to the set of instances that remains after pruning the rule. This solves the main problem of the global approach to pruning. However, there is also a drawback to this "local" pruning procedure: it does not know about potential new rules when it prunes the current rule; the pruning decisions are based on the accuracy of the current rule only. In other words, the algorithm cannot estimate how pruning the current rule will affect the generation of future rules. Simply put, IREP prunes "blindly" without any knowledge of likely developments in further cycles of the separate-and-conquer algorithm. This myopia has the consequence that it can suffer from overpruning. First, it can prune too many conditions off the current rule; second, it can stop generating further rules too early (Cohen, 1995). Both phenomena result in an overall loss of accuracy. We first discuss the former type of pathology, before proceeding to the latter one.

As the following example shows, the basic strategy of building a single rule and pruning it back by deleting conditions can lead to a problematic form of overpruning, which we call "hasty generalization."[2] This is because the pruning interacts with the

[2]This apt term was suggested by an anonymous referee for Frank and Witten (1998a).

Rule	Coverage			
	Training Set		Pruning Set	
	A	B	A	B
1: a = true $\Rightarrow A$	90	8	30	5
2: a = false $\wedge$ b = true $\Rightarrow A$	200	18	66	6
3: a = false $\wedge$ b = false $\Rightarrow B$	1	10	0	3

Figure 5.5: A hypothetical target concept for a noisy domain.

covering heuristic. Generalizations are made before their implications are known, and the covering heuristic then prevents the learning algorithm from discovering the implications.

Here is a simple example of hasty generalization. Consider a Boolean dataset with attributes a and b built from the three rules in Figure 5.5, corrupted by ten percent class noise. Assume that the first rule has been generated and pruned back to

$$a = true \Rightarrow A$$

(The training data in Figure 5.5 is there solely to make this scenario plausible.) Now consider whether the rule should be further pruned. Its error rate on the pruning set is 5/35, and the null rule

$$\Rightarrow A$$

has an error rate of 14/110, which is smaller. Thus the rule set will be pruned back to this single, trivial, rule, instead of the patently more accurate three-rule set shown in Figure 5.5. Note that this happens because the algorithm concentrates on the accuracy of rule 1 when pruning—it does not make any guesses about the benefits of including further rules in the classifier.

Hasty generalization is not just an artifact of pruning with a hold-out set: it can also happen with other underlying pruning mechanisms, for example, C4.5's error-based pruning (Quinlan, 1992). Because of variation in the number of noisy instances in the data sample, one can always construct situations in which pruning causes rules with comparatively large coverage to swallow rules with smaller but significant coverage. This can happen whenever the number of errors committed by a rule is large compared with the total number of instances covered by rules that are adjacent in instance space.

The second reason for overpruning, early stopping, occurs because the algorithm is designed to concentrate on the positive examples first, leaving the negative ones to be covered by the default rule. Because of this strategy, IREP can get trapped into generating a spurious rule for the positive class that turns out to have an error rate greater than 0.5. Consequently it will stop producing any further rules even though there might be other, unexplored areas of the instance space for which positive rules are appropriate. These areas will be classified incorrectly as belonging to the negative class by the final rule set.

The problem of early stopping can be solved by generating a proper decision list that allows for rules of both classes to occur anywhere in the rule set. However, it is not easy to modify the basic IREP procedure to produce an accurate decision list. In principle, a decision list can be generated by evaluating each rule with respect to its majority class. This involves replacing the concentration measure above by simple accuracy,

$$accuracy = \frac{\max(p, n)}{p + n},$$

and adjusting the information gain correspondingly. Ironically this modification vastly increases the likelihood of hasty generalization because pruning based on accuracy is much more aggressive than pruning based on the concentration measure. Consequently this change leads to severe underpruning, the very problem it is supposed to combat. The next section presents a new algorithm, based on IREP, that generates a decision list and avoids hasty generalization as well as early stopping.

Note that Cohen (1995) has proposed a solution to early stopping that replaces the simple error-based stopping criterion by a procedure based on the minimum description length principle. His method allows IREP to jump out of a "local minimum" in search space by imposing a less restrictive stopping rule. Unfortunately this also means that IREP can generate many spurious rules in the process. Therefore, rule induction must be followed by a global optimization step to simplify and adjust the initial rule set (Cohen, 1995).

5.1.3 Incremental tree-based pruning

The main problem of IREP is caused by the fact that it bases pruning decisions on the current rule only. It does not take into account the impact of these decisions on prospective rules that can improve the classifier's overall performance. This section presents an approach to incremental rule learning using decision tree structures that is designed to overcome this myopia. It is inspired by the standard method of deriving rule sets from decision trees employed by the rule learner included in C4.5 (Quinlan, 1992).

This standard strategy begins by creating an unpruned decision tree. Subsequently it transforms the tree into a rule set by generating one rule for each path from the root to a leaf. Most rule sets derived in this way can be simplified dramatically without sacrificing predictive accuracy. They are unnecessarily complex because the disjunctions that they imply can often not be expressed succinctly in a decision tree. This is the replicated subtree problem discussed in the introduction to this chapter. Consequently C4.5 performs global optimization to simplify the initial rule set. Although this process produces accurate rule sets that are often more accurate than the initial decision tree, it is complex and time-consuming. It has been shown that for noisy datasets, runtime is cubic in the number of instances (Cohen, 1995). Moreover, despite the lengthy optimization process, rules are still restricted to conjunctions of those attribute-value tests that occur along a path in the initial decision tree.

The incremental separate-and-conquer technique employed by IREP does not suffer from these problems. However, it is handicapped by the potential for hasty generalization. The key idea for overcoming this obstacle is to combine the two main paradigms for rule learning: the separate-and-conquer method on the one hand, and the strategy of obtaining a rule set from a decision tree on the other. This leads to the new rule induction algorithms presented in this chapter.

Incremental pruning with trees

The basic form of the resulting algorithm, depicted in Figure 5.6, is very simple. It is identical to the separate-and-conquer algorithm from Figure 5.3 except for the way in which a single rule is derived before being added to the rule set. The

```
Procedure TB-IREP(D: training data)

R := empty set of rules
while not D empty
   split D into growing set and pruning set
   build decision tree on growing set and prune on pruning set
   r := best rule from decision tree
   R := add r to R
   remove instances from D that are covered by r
return R
```

Figure 5.6: Tree-based incremental reduced-error pruning

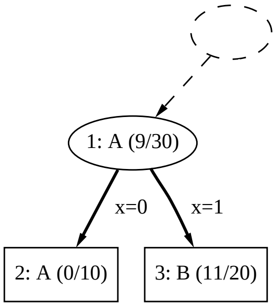
```
Procedure PruneTree(T: tree, D: pruning data)

if T is leaf
   return
for all branches of T
   prune tree attached to branch
e := error rate of T's root
e' := minimum error rate among all of T's leaves
if e ≤ e'
   replace tree by leaf
return
```

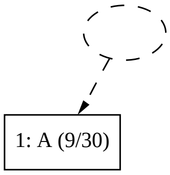
Figure 5.7: Decision tree pruning in TB-IREP

new algorithm adopts IREP's strategy of adding a *pruned* rule at each step of the separate-and-conquer scheme. This avoids the problems of the global pruning approaches discussed above. However, the procedure it uses to generate this pruned rule differs from the method employed by IREP. Whereas IREP grows a single rule and prunes it, TB-IREP builds a full decision tree, prunes it, and selects a single rule from it.

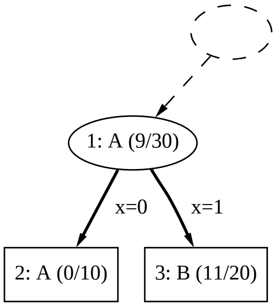
The unpruned decision tree is grown using an ordinary decision tree inducer, for example, Quinlan's C4.5 tree learner (Quinlan, 1992), which performs multiway splits on nominal attributes and binary splits on numeric ones, chosen to maximize an entropy-based measure. Subsequently the fully expanded tree is pruned in a bottom-up fashion using pruning data that was held out when the tree was grown. An important feature of the pruning algorithm is that it is optimized for the final objective of obtaining an accurate rule. Therefore it differs from standard reduced-error pruning for decision trees.



(a) Unpruned subtree



(b) Standard reduced-error pruning



(c) TB-IREP pruning

Figure 5.8: Example illustrating the two pruning procedures

The pruning algorithm is summarized in Figure 5.7. Like standard reduced-error pruning for decision trees (Quinlan, 1987a), discussed in Section 3.1, it traverses the tree in a recursive bottom-up fashion and estimates performance using the pruning data. However, it differs from standard reduced-error pruning in the way pruning decisions are made. Whereas the standard procedure replaces a subtree by a leaf whenever this does not increase the number of incorrect classifications that the tree produces, the new procedure replaces a subtree by a leaf node X only if none of the subtree's leaves exhibits a lower error rate than X. The rationale behind the new operation is that every leaf of the decision tree is a potential rule that could be included in the rule set produced by TB-IREP. Replacing a subtree is the same as pruning the tails of all rules corresponding to the subtree's leaves. Thus it is desirable that pruning only proceeds if this does not reduce the accuracy of any of the rules involved. This is exactly what the algorithm in Figure 5.7 achieves.

Figure 5.8 shows a two-class example that illustrates the difference between the two pruning schemes. The initial unpruned subtree in Figure 5.8a has two leaves, one classifying ten instances in the pruning data correctly as class A, the other classifying nine out of 20 correctly as class B. Standard reduced-error pruning would replace the unpruned subtree by the leaf shown in Figure 5.8b because this would decrease the total number of misclassifications—the unpruned subtree misclassifies 11 instances and the leaf node only nine. In contrast, the modified TB-IREP pruning operation would abstain from pruning (Figure 5.8c) because one of the two leaves has an error rate of 0, which is lower than the root node's error rate of 9/30.

Building and pruning a decision tree to obtain a rule solves both of IREP's overpruning problems. First, it prevents hasty generalization because all the implications are known when a subtree is pruned. The pruning algorithm considers a set of neighboring rules and prunes only if this does not negatively affect the performance of any of these prospective rules. Second, early stopping is no longer an issue because there is no need for a stopping criterion. TB-IREP will only terminate when the pruning algorithm in Figure 5.7 prunes the tree structure right back to its root node. This event implies that no additional structure can be found in the data remaining from the previous separate-and-conquer iterations. The pruned decision tree works as an approximation to potential future rules. If this approximation collapses to a single leaf, the algorithm terminates. In that case the root node will become the default rule and TB-IREP will stop.

One step of the pseudo-code for TB-IREP in Figure 5.6 is yet to be defined, namely the part that selects a rule from the pruned decision tree. Once the pruning process has terminated, one of the leaves in the tree must be selected, so that the corresponding rule can be appended to TB-IREP's rule set. The discussion of the pruning process immediately suggests a criterion for selecting an appropriate rule: the leaf with the highest accuracy on the pruning data. By definition this criterion always selects a leaf that has prevented further pruning of the decision tree. This means that, according to the performance on the pruning data, it is better not to prune the rule any further.

Time complexity

It is obvious that TB-IREP is slower than IREP: the former must build a full pruned decision tree to obtain a single rule, whereas the latter can generate the rule directly. Moreover, because TB-IREP does not suffer from hasty generalization and early stopping, it often produces more rules than IREP. However, the asymptotic difference between building a full decision tree and a single rule is relatively small.

Above it is shown that IREP can generate a pruned rule in time $O(n)$, where n is the number of training instances. In the following we show that the time complexity for building and pruning TB-IREP's decision tree is $O(n \log n)$.

For simplicity, assume that each node splits the training data into two subsets A and B, the larger of which contains at most $(100/b)\%$ of the instances, and that b is bounded away from one. The second assumption is similar to the one made in the derivation for the time complexity of building a single rule. Given these assumptions the tree has depth $O(\log_b n_{grow}) = O(\log n_{grow})$, where n_{grow} is the number of instances in the growing set. Furthermore, at most n_{grow} instances are processed at each of the tree's $O(\log n_{grow})$ levels. This means that the overall time complexity for growing a tree is

$$O(n_{grow} \times \log n_{grow}) = O(n \log n).$$

However, the pruning step must also be taken into account. First, the pruning data must be disseminated in the tree structure so that the error rate for each node can be calculated. As in the growing phase, this can be done in time $O(n_{prune} \times \log n_{prune})$, where n_{prune} is the number of instances in the pruning data. Then the bottom-up pruning procedure must be invoked. The pseudo-code of Figure 5.7 suggests that this procedure must visit most of the tree's nodes several times because it must determine the minimum error rate occuring at a leaf of each subtree considered for replacement. Fortunately this can be avoided and the whole pruning process can proceed in one pass where each node is visited only once. This is a direct consequence of the fact that the minimum error rate e' for a subtree is the smallest of the minimum error rates for all its branches. If this minimum is too high, the subtree is replaced by a leaf; otherwise e' is passed to the node the subtree is attached to. This means that the minimum error rate for a subtree can be calculated on the fly by looking at the error rates associated with its branches. Hence each node of the tree can be visited only once, namely when the corresponding subtree is considered for replacement. Because there are $O(n_{grow})$ nodes in the subtree, the overall time complexity for pruning the decision tree is

$$O(n_{prune} \times \log n_{prune}) + O(n_{grow}) = O(n \log n).$$

Because the time complexities for building and pruning the tree are both $O(n \log n)$, it follows that the overall time complexity for generating the pruned tree is $O(n \log n)$. Rule selection can be done in time linear in the number of instances because there are $O(n_{grow}) = O(n)$ leaves in the tree from which to choose the most accurate rule. Thus TB-IREP needs time $O(n \log n)$ to complete one cycle of the separate-and-conquer scheme. Although this is slower than IREP, the difference is only a log factor and therefore not problematic for all but very large datasets. The theoretical worst-case time complexity is $O(n^2 \log n)$, and occurs when the number of generated rules increases linearly with the number of training instances.

5.2 Improved rule sets with significance tests

TB-IREP as described above chooses the most accurate leaf from the decision tree for inclusion in the rule set. However, it is not clear whether this is the best possible choice (Witten & Frank, 2000). Consider, for example, a leaf A that covers three pruning instances, classifying them all correctly, and another leaf B that covers 100 instances, misclassifying only one of them. Although A is more accurate than B on the pruning data, B is clearly the better rule. There are two reasons for this. First, B is likely to have an error rate very close to zero on fresh data, whereas the good performance of rule A might be just a fluke. There is no guarantee that A will achieve a low error rate on new data. Second, rule B will be much more interesting for the user because it makes a far more general statement about the domain. Hence it makes sense to take the statistical significance of a leaf into account when a rule is selected from the decision tree.

5.2.1 Using statistical significance for rule selection

There is a natural definition of the statistical significance of a rule: the probability that an equally or more accurate rule with the same generality can be obtained by chance alone (Gaines, 1989). This probability is the p-value of a permutation test on the corresponding 2×2 contingency table, depicted in Figure 5.9. The table summarizes the rule's performance on the pruning data. In this table, TP (FP) denotes the number of "true positives" ("false positives"), that is, the number of

	TP+FP	FN+TN
P	TP	FN
N	FP	TN

Figure 5.9: Contingency table representing a rule

	TP+FP	FN+TN	
P	3	497	500
N	0	500	500
	3	997	1000

$p_F = 0.125$

(a)

	TP+FP	FN+TN	
P	100	400	500
N	1	499	500
	101	899	1000

$p_F = 1.81 \times 10^{-31}$

(b)

Figure 5.10: Contingency table representing two example rules

instances covered by the rule that it classifies correctly (incorrectly). Analogously, TN (FN) denotes the number of "true negatives" ("false negatives"), in other words, the number of instances not covered by the rule that would be classified incorrectly (correctly) if they were covered by it. A good rule maximizes TP while minimizing FP.

The probability of observing an equally or more accurate rule of the same generality is the same as the probability of observing a hypothetical contingency table with the same marginal totals for which TP is at least as large. This is exactly what is measured by the p-value p_F of the one-sided version of Fisher's exact test (Good, 1994):

$$p_F = \sum_{x=0}^{\min(FP,FN)} \frac{\binom{TP+FP}{TP+x} \times \binom{FN+TN}{FN-x}}{\binom{P+N}{P}} \tag{5.1}$$

The lower the p-value of this test for a given rule, the more statistically significant it is.

Figure 5.10 shows the contingency tables for the two example rules from the beginning of this section, assuming that the pruning data contains a total of 1000 instances, 500 of which belong to the class predicted by the two rules. The figure also contains the p-values for the two tables: 0.125 for that in Figure 5.10a and 1.81×10^{-31} for Figure 5.10b. This means that there is a greater than 10% chance of observing a rule that is as general as the first example rule and at least as accurate, while for the second rule this probability is extremely low.

There is one caveat when rules are selected from the decision tree by considering their statistical significance: the most significant leaf might represent a rule that is less accurate on the pruning data than one of its potential prunings. We want the algorithm to choose among only those rules that prevent further pruning of the decision tree. Therefore we introduce the notion of the "eligibility" of a node in the decision tree.

Definition 1. *A node is "eligible" for selection if and only if*

1. *It is more accurate than all its prefixes.*

2. *None of its successors is eligible.*

Note that condition 2 can only be fulfilled for leaf nodes of the decision tree pruned by TB-IREP. That means only leaf nodes of the pruned decision tree can be eligible. Condition 1 can easily be checked for each leaf once the pruning process is complete, which can be done in time linear in the number of nodes. Finally, the most significant eligible leaf is chosen for inclusion in the rule set.

However, rule selection is not the only step in the TB-IREP algorithm where significance testing is appropriate. It can also be useful for pruning. TB-IREP's pruning operation is very conservative because it only replaces a subtree by a leaf if this does not decrease the accuracy of any of the affected rules on the pruning data. As explained above, this is an efficient means of preventing hasty generalization and early stopping—phenomena that can cause IREP to discard truly predictive parts of the model. However, there is a downside to this procedure: it can overfit the pruning data on noisy datasets. Consequently it may generate an overly complex model when it could choose a simpler, more comprehensible, and potentially more accurate description of the data.

5.2.2 Examples for underpruning

Figure 5.11a shows an example where TB-IREP's pruning procedure is likely to degrade comprehensibility without improving accuracy. In this subtree, both leaf nodes assign all instances to class A. However, node 2 does so with slightly higher (observed) accuracy than node 3. It classifies eight of ten instances in the pruning data correctly; node 3 classifies only seven of ten instances correctly. Because node 2 has higher accuracy than node 1, TB-IREP will abstain from making node 1 a

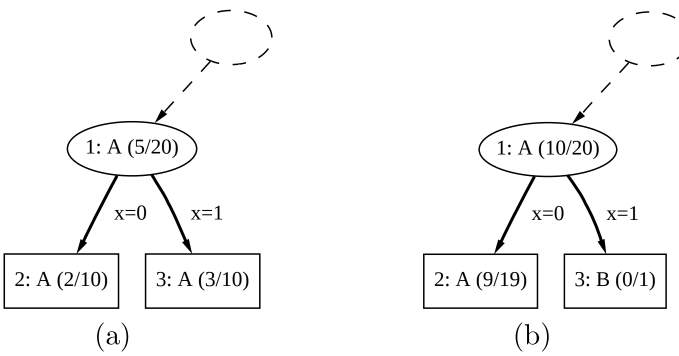


Figure 5.11: Two examples where TB-IREP does not prune sufficiently

leaf node. Consequently, if node 2 is selected for inclusion in the rule set, the rule is (a) unnecessarily complex and (b) the instances covered by node 3 are likely to lead to an additional rule in subsequent iterations of the separate-and-conquer scheme. However, the difference between nodes 2 and 3 is only marginal and most likely due to sampling variation. Hence it would be beneficial to turn the subtree into a leaf.

However, comprehensibility is not the only issue. Consider the example subtree in Figure 5.11b, which also contains two leaves. In this case node 2 belongs to class A and node 3 to class B. Again TB-IREP would not prune this subtree because both leaves are more accurate than node 1: node 2 classifies 10 of 19 instances in the pruning data correctly and node 3 assigns one instance correctly to class B. However, the difference between nodes 1 and 2 is very small. It is quite likely that the favorable distribution of pruning instances at node 3 is due to chance alone. Unfortunately it is possible that TB-IREP will select node 3 for inclusion in the rule set because it can have a lower p-value than node 2—for example, if the remainder of the tree assigns one instance correctly to class A. In this example it is likely that accuracy as well as comprehensibility would improve if node 1 were made into a leaf.

The problem in both examples is that pruning does not take the statistical significance of the distribution of pruning instances into account. In other words, it does not test whether the observed association between the model and the data is due solely to random fluctuations in the sampling process or whether it is a genuine feature of the domain. Fortunately there exist well-founded off-the-shelf tools for this kind of model validation that are used throughout this thesis: statistical significance tests on contingency tables. In Chapter 3, for example, they are used in the similar context of reduced-error pruning for decision trees. However, it is not possible

	TP+FP	TN+FN	
P	8	7	15
N	2	3	5
	10	10	20

$p_F = 0.5$

	TP+FP	TN+FN	
P	7	8	15
N	3	2	5
	10	10	20

$p_F = 0.848$

(a)

	TP+FP	TN+FN	
P	10	0	10
N	9	1	10
	19	1	20

$p_F = 0.5$

	TP+FP	TN+FN	
P	1	9	10
N	0	10	10
	1	19	20

$p_F = 0.5$

(b)

Figure 5.12: Contingency tables for the examples from Figure 5.11

to apply the procedure from Chapter 3 directly, because pruning in TB-IREP is different from standard reduced-error pruning in decision trees. It is not the validity of a subtree as a whole that matters; rather, it is the individual performance of each of its leaves.

5.2.3 Integrating significance testing into pruning

It is straightforward to integrate significance testing into TB-IREP. The modified algorithm is based on the observation that a rule should only be pruned if none of its extensions is more accurate in a statistically significant sense. We consider an extension to be statistically significant if it is unlikely that an equally general extension of the same accuracy can be obtained by chance alone. This is exactly what is measured by the p-value of the one-sided version of Fisher's exact test, discussed above.

Figure 5.12 depicts the contingency tables corresponding to each of the leaves in Figure 5.11. It also shows the p-value for each table according to Fisher's exact test, denoted by p_F. As it turns out, none of the leaves is statistically significant even at the 30% level. Thus, if TB-IREP includes this test in the pruning procedure with a sufficiently low significance level, it will successfully avoid the problems discussed above.

In the following we explain in detail how significance testing is integrated into the TB-IREP pruning procedure. The modified algorithm, displayed in Figure 5.13, is based on the definition of the "statistical eligibility" of a rule, which replaces simple

```
Procedure PruneTreeWithSigTest(T: tree, D: pruning data)

if T is leaf
   return
for all branches of T
   prune tree attached to branch
Check significance of each extension with respect to T's root
Delete subtrees that do not contain significant extensions
e := error rate of T's root
e' := minimum error rate among all of T's significant leaves
if no significant leaf can be found or e ≤ e'
   replace tree by leaf
return
```

Figure 5.13: Decision tree pruning with significance testing in TB-IREP

error-based eligibility defined above:

Definition 2. *A node is "statistically eligible" for selection if and only if*

1. *It is more accurate than all its prefixes.*

2. *It is a significant extension of all its prefixes.*

3. *None of its successors is eligible.*

To implement this definition TB-IREP must check the significance of each extension with respect to each node in the tree, because leaves that are significant extensions of nodes in lower parts of the tree can be insignificant extensions of nodes higher up in the tree. Consequently subtrees in lower parts of the tree that have already been considered for pruning may have to be turned into leaves because they no longer contain nodes that are statistically significant.

The minimum error rate is computed from the *significant* leaves only. If the minimum error rate of all significant leaves is at least as large as the error rate of the subtree's root node, the subtree is converted into a leaf. The subtree is also replaced if no significant leaves can be found. Once the pruning algorithm has terminated, rule selection can be performed in the same way as before. The only difference is that only statistically eligible rules can be included in the rule set. In other words, the maximally significant rule is chosen from those leaves that are statistically eligible.

A potential problem is the choice of the significance level. The optimum significance level—leading to the smallest rule set that achieves maximum accuracy on fresh data—depends on the properties of the domain, as discussed in Chapter 3. The problem is analogous to the situation encountered in Chapter 3, where the right significance level for applying significance tests in decision tree pruning is found using cross-validation. The same procedure can be applied here. For each significance level, accuracy is estimated by cross-validation. Then the significance level that maximizes this estimate is chosen, and the rule set rebuilt from the full training data by applying this optimized value.

Cross-validation is relatively slow because the pruned models are not nested as in the case of decision trees. An alternative is to choose the significance level depending on the amount of data that is available (Martin, 1997). A low significance level is used if large quantities of data are available for pruning, and a higher level if little data is present. This heuristic is based on the observation that significance testing is very unlikely to find a significant association if data is sparse. In this case, it is often better to perform little or no pruning (Fisher, 1992).

Significance testing slightly increases the asymptotic time complexity of learning a rule with TB-IREP. Computing the p-value of Fisher's exact test for all nodes of the same depth is $O(n)$ because it is never worse than iterating through all the pruning instances present at that level of the tree. In the worst case, all the nodes at each level must be evaluated with respect to all their predecessors. Thus, because there are at most $O(\log n)$ levels in the tree and each node has at most $O(\log n)$ prefixes, the time complexity for computing the statistical eligibility of each node is $O(n \log^2 n)$. This increases the time complexity for generating a rule slightly from $O(n \log n)$. However, it is likely that the increase is offset by the fact that the more stringent pruning procedure will generate fewer rules.

5.3 Rule learning with partial trees

Section 5.1.3 shows that the extra cost of building a decision tree instead of an individual rule is only $O(\log n)$ for error-based pruning. Nevertheless, it seems wasteful to build a full tree to obtain a single rule. TB-IREP effectively discards the information in the tree once it has read off a rule. The additional time complexity is certainly noticeable when large datasets are processed. The decision tree's main purpose is to avoid the overpruning phenomenon that occurs when a single rule is pruned. A more economical approach would only build those parts of the tree that are relevant for pruning the final rule. It turns out that this is indeed feasible with a minor change in TB-IREP's basic pruning procedure.

5.3.1 Altering the basic pruning operation

In the pruning algorithm of Figure 5.5 every subtree is a potential candidate for replacement. This is equivalent to considering every tail of the corresponding rules for deletion. However, it is possible to adopt a slightly more restrictive policy where only the last condition of a rule is considered for deletion (Fürnkranz & Widmer, 1994). Translated into the context of a decision tree, this means that a subtree is only considered for pruning if, apart from the subtree's root node, all other nodes are leaves. In the case of error-based pruning, the subtree is replaced by a leaf node if its root node is at least as accurate as all its leaves. This pruning procedure is based on the following definition of eligibility, which is a modification of the definition in Section 5.2.1.

Definition 3. *A node is "PART-eligible" for selection if and only if*

1. *It is more accurate than its immediate prefixes.*

2. *None of its successors is eligible.*

It is also possible to use significance-based pruning in this restricted fashion. This means that the subtree is replaced by a leaf node if it is at least as accurate as its significant extensions. In that case, the following definition of eligibility applies—a modification of the definition from Section 5.2.3.

Definition 4. *A node is "statistically PART-eligible" for selection if and only if*

1. *It is more accurate than its immediate prefixes.*

2. *It is a statistically significant extension of its immediate prefixes.*

3. *None of its successors is eligible.*

Procedure ExpandSubset(G: growing data, P: pruning data)

select best split using G
split G into subsets G_i and P into subsets P_i
while there are subsets G_i that have not been expanded **and**
    all subsets G_i expanded so far are leaves
  G_m := subset with minimum entropy that has not been expanded
  call ExpandSubset(G_m,P_m)
if all the expanded subsets G_i are leaves
  e := error rate for node on P
  e' := minimum error rate among all leaves according to P_i
  if $e \leq e'$
    replace tree by leaf
return

Figure 5.14: Method that expands a given set of instances into a partial tree

Of course, this modified pruning procedure is still recursive, that is, by deleting a subtree the subtree rooted at the next higher level may become a candidate for replacement. Because the new replacement operation considers only a subset of the pruning options evaluated by the old one, it is possible that, at first, it may overlook a potentially useful simplification. However, due to the recursive fashion in which the algorithm proceeds, it may recover this solution at a later stage. Only experiments on practical datasets can determine whether the new operation makes any difference in real-world applications.

Modifications of the pruning procedure do not change the time needed for building the decision tree. Thus the new pruning operation has no direct impact on TB-IREP's overall time complexity. The key to a significant speed-up of the overall learning process is that the new pruning operation allows the actual tree inducer to be modified so that a rule can be obtained much faster. The basic idea is to generate only those parts of the decision tree that are relevant for pruning the final rule. The resulting rule learning algorithm is called "PART" because it generates "partial decision trees"—decision trees that contain branches to undefined subtrees. To generate such a tree, we integrate the construction and pruning operations in order to find a "stable" subtree that can be simplified no further. Once this subtree has been found, tree-building ceases and a single rule is read off.

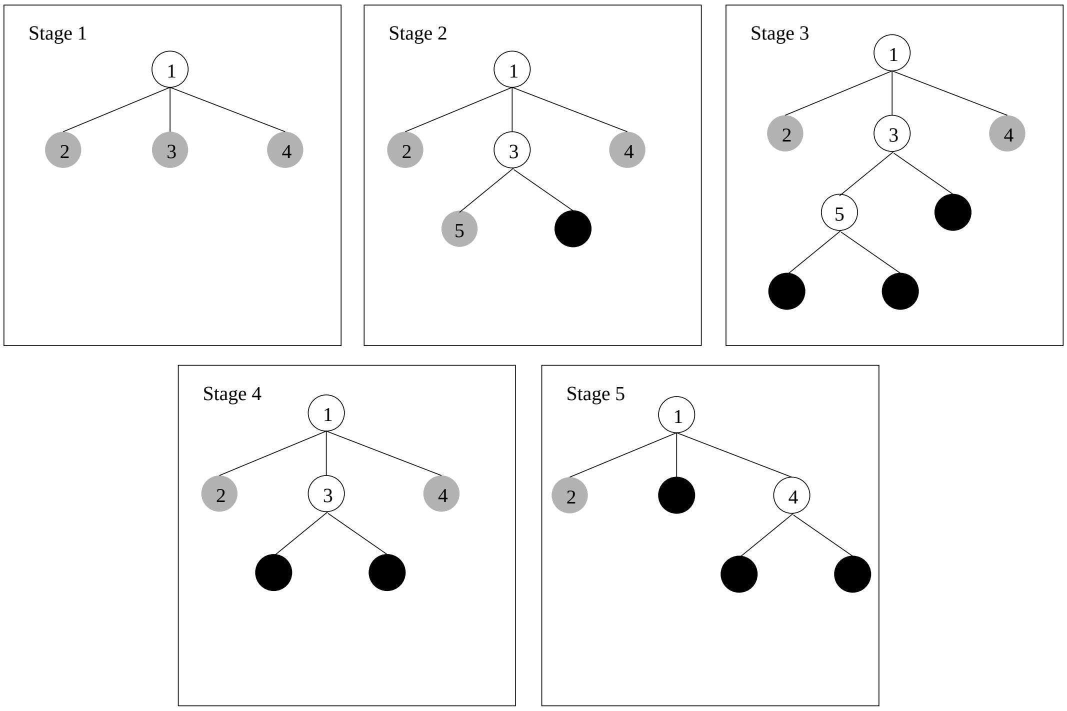


Figure 5.15: Example of how a partial tree is built

5.3.2 Building a partial tree

The tree-building algorithm is summarized in Figure 5.14. It splits a set of instances recursively into a partial tree. The first step chooses a test and divides the instances into subsets accordingly. This is just the standard procedure in a decision tree inducer. Then the subsets are expanded in order of their average entropy, starting with the smallest. The reason for this is that a subset with low average entropy is more likely to result in a small subtree and therefore produce a more general rule. This continues recursively until a subset is expanded into a leaf, and then continues further by backtracking. But as soon as an internal node appears that has all its children expanded into leaves, pruning begins: the algorithm checks whether that node is better replaced by a single leaf. This is where the new subtree replacement operation from above comes into play. If replacement is performed the algorithm backtracks in the standard way, exploring siblings of the newly-replaced node.

However, if during backtracking a node is encountered all of whose children are not leaves—and this will happen as soon as a potential subtree replacement is *not* performed—then the remaining subsets are left unexplored and the corresponding subtrees are left undefined. This means that a subtree has been found that cannot be pruned any further. Due to the recursive structure of the algorithm this event automatically terminates tree generation. Because each leaf corresponds to a prospective rule, this implies that the current set of expanded rules cannot be pruned any further.

Figure 5.15 shows a step-by-step example. During stages 1–3, tree-building continues recursively in the normal way—except that at each point the lowest-entropy sibling is chosen for expansion: node 3 between stages 1 and 2. Gray nodes are as yet unexpanded; black ones are leaves. Between stages 2 and 3, the black node will have lower entropy than its sibling, node 5; but cannot be expanded further since it is a leaf. Backtracking occurs and node 5 is chosen for expansion. Once stage 3 is reached, there is a node—node 5—that has all of its children expanded into leaves, and this triggers pruning. Subtree replacement for node 5 is considered, and accepted, leading to stage 4. Now node 3 is considered for subtree replacement, and this operation is again accepted. Backtracking continues, and node 4, having lower entropy than 2, is expanded—into two leaves. Now subtree replacement is considered for node 4: let us suppose that node 4 is not replaced. At this point, the process effectively terminates with the 3-leaf partial tree of stage 5.

5.3.3 Remarks

Like TB-IREP's more expensive pruning method, this procedure ensures that the overpruning effect discussed in Section 5.1.2 cannot occur. A node can only be pruned if all its successors are leaves. This can only happen if all its subtrees have been explored and either found to be leaves, or are pruned back to leaves. Thus all the implications are known when a pruning decision is made. Situations like that shown in Figure 5.5 are therefore handled correctly. Note that the trick of building and pruning a partial tree instead of a full one is only possible because of the more restrictive pruning operation introduced in the beginning of this section. If every subtree of a tree was considered for replacement—instead of only those ones that are decision stumps—the procedure would inevitably have to build the full decision tree.

Once a partial tree has been built, a single rule is extracted from it. This is done in the same way as in TB-IREP. The only difference is that the partial tree contains only a subset of the leaves that appear in the fully expanded tree considered by

TB-IREP. Each PART-eligible leaf is a candidate for inclusion in the rule set, and we seek the most accurate, or most significant, PART-eligible leaf of those subtrees that have been expanded into leaves.

If a dataset is noise-free and contains enough instances to prevent the algorithm from doing any pruning, just one path of the full decision tree is explored. This achieves the greatest possible performance gain over the naive method that builds a full decision tree each time. It is equivalent to generating a single rule. As shown in Section 5.1.2, this can be done in time linear in the number of training instances. The gain decreases as more pruning takes place. The runtime is bounded above by the time it takes TB-IREP to generate a single rule. As explained in Section 5.1.3, this upper bound contains an extra factor proportional to the logarithm of the number of training instances when purely error-based pruning is applied. In the case of significance-based pruning, the worst-case time complexity for PART is lower than for TB-IREP because significance testing is only done with respect to the immediate predecessors of a node, rather than all of them. This means that the additional log factor required for TB-IREP does not appear in the computation of the time complexity for PART. Hence, for PART, the time complexity for purely error-based and significance-based pruning is the same.

5.4 Experiments

This section empirically investigates the performance of the rule learning algorithms discussed above. First, standard incremental reduced-error pruning is compared to tree-based incremental reduced-error pruning, both with respect to the accuracy and the size of the induced rule sets. Then the effect of integrating significance testing into the rule learning mechanism is discussed. Finally we present results for generating partial decision trees instead of full ones.

5.4.1 Tree-based and standard incremental reduced-error pruning

TB-IREP is designed to prevent hasty generalization and early stopping, and is therefore likely to produce more accurate but also larger rule sets than IREP. This section verifies empirically whether this is really the case on practical datasets.

The experimental methodology is the same as in previous chapters. Ten-fold cross-validation repeated ten times is used to estimate both accuracy and size of the induced rule sets. A difference in the estimates is considered significant if it is statistically significant at the 5%-level according to a paired two-sided t-test on the ten data points obtained from the ten cross-validation runs.

For both TB-IREP and IREP, exactly the same random, stratified splits are used to divide the original training data into growing and pruning sets. TB-IREP requires a method for choosing a split on a nominal or numeric attribute in order to generate a tree structure from the growing data. Our implementation makes this choice in exactly the same way as Release 8 of the decision tree inducer C4.5 (Quinlan, 1992). Experiments on practical datasets also necessitate a method for dealing with missing values. In our implementations of both TB-IREP and IREP we adopt a very simple policy: all missing values are replaced by the mode or mean observed from the training data.

Some of the datasets contain more than two classes. In order to apply IREP to multi-class problems we use the same procedure as Cohen (1995). First, the m classes are ordered according to the number of training instances pertaining to each class. Let this ordered list of classes be $c_1, c_2, \ldots, c_m$. Then IREP is applied to learn a rule set distinguishing class c_1 from all the other classes. After that, the instances covered by the rule set are removed, and the whole process is repeated for classes $c_2, \ldots, c_{m-1}$. Then the resulting individual rule sets are concatenated into one decision list in the order in which they were generated. Finally a default rule is appended that assigns all instances to the remaining, most populous, class c_m. This produces a decision list where the rules for each class appear consecutively.

Table 5.1 shows the estimated accuracies for both TB-IREP and IREP. A $\bullet$ ($\circ$) marks a dataset where TB-IREP performs significantly better (worse) than IREP. The results for the significance tests are summarized in Table 5.2, which shows that TB-IREP is significantly more accurate than IREP on twelve datasets, and significantly less accurate on only two (breast-cancer and horse-colic). This confirms the first hypothesis of this chapter: TB-IREP generally produces more accurate rule sets than IREP. On some datasets the difference in accuracy is quite large. It spans, for example, more than 10% on the audiology, vehicle, and vowel datasets.

Table 5.1: Accuracy and number of rules for TB-IREP compared to IREP

	Accuracy			Number of rules		
Dataset	TB-IREP	IREP		TB-IREP	IREP	
anneal	98.3±0.4	97.9±0.3	•	13.8±0.4	8.1±0.5	∘
audiology	76.5±1.1	63.3±2.8	•	20.0±0.3	12.6±0.6	∘
australian	84.5±1.2	84.4±1.1		37.1±1.2	5.3±0.4	∘
autos	72.0±2.2	64.5±3.2	•	23.1±0.8	6.9±0.5	∘
balance-scale	81.4±1.0	76.1±1.8	•	36.1±1.0	7.4±0.5	∘
breast-cancer	68.3±2.3	72.6±0.8	∘	29.6±1.1	2.4±0.2	∘
breast-w	94.8±0.6	94.7±0.5		11.8±0.9	4.7±0.4	∘
german	71.0±0.7	71.6±0.8		71.3±2.4	3.4±0.4	∘
glass-2	77.9±2.4	76.8±3.7		9.1±0.7	3.9±0.3	∘
glass	69.0±2.5	62.6±3.1	•	15.6±0.6	5.7±0.2	∘
heart-c	78.5±1.6	79.1±1.7		19.8±1.4	3.7±0.2	∘
heart-h	80.3±1.6	79.9±1.6		18.5±0.8	3.0±0.3	∘
heart-statlog	78.9±1.5	77.9±1.5		16.2±0.9	3.5±0.2	∘
hepatitis	79.5±2.6	79.7±2.1		9.1±0.5	1.6±0.2	∘
horse-colic	80.9±1.8	83.6±1.2	∘	21.8±0.7	3.0±0.3	∘
ionosphere	90.2±1.0	88.1±1.1	•	9.6±0.6	3.8±0.2	∘
iris	94.9±0.9	94.0±0.9		4.0±0.2	3.4±0.2	∘
labor	83.9±4.7	79.0±4.3	•	3.5±0.4	2.6±0.3	∘
lymphography	76.0±2.3	74.7±3.6		11.2±0.4	4.3±0.3	∘
pima-indians	72.8±1.2	73.6±1.3		21.8±1.8	3.6±0.4	∘
primary-tumor	39.0±1.0	35.2±1.2	•	44.4±1.9	5.3±0.4	∘
sonar	73.1±1.7	72.8±1.6		10.7±0.8	2.8±0.3	∘
soybean	91.9±0.4	88.5±0.7	•	36.9±1.1	23.4±0.5	∘
vehicle	72.6±1.0	61.3±2.8	•	42.5±3.0	10.2±0.8	∘
vote	95.1±0.6	94.8±0.7		8.1±0.6	3.5±0.1	∘
vowel	77.6±0.9	56.0±2.6	•	70.2±1.9	26.1±1.1	∘
zoo	90.0±3.0	86.3±1.3	•	7.3±0.3	6.0±0.2	∘

Table 5.2: Results of paired t-tests (p=0.05): number indicates how often method in column significantly outperforms method in row

	Accuracy		Number of rules	
	TB-IREP	IREP	TB-IREP	IREP
TB-IREP	–	2	–	27
IREP	12	–	0	–

Table 5.1 also includes the size of the corresponding rule sets. A $\bullet$ ($\circ$) marks a dataset where TB-IREP produces significantly fewer (more) rules than IREP, and Table 5.2 summarizes the results of the significance tests. IREP produces significantly fewer rules than TB-IREP on all 27 datasets included in the experiment. On some datasets the difference in the number of rules is very large. On the german dataset, for example, TB-IREP produces 20 times more rules than IREP.

These findings show that IREP produces very small rule sets. However, on many datasets this is a result of overpruning, which seriously affects the rule sets' accuracy. TB-IREP, on the other hand, successfully prevents overpruning, and produces more accurate rule sets than IREP on many datasets. However, the results also show that TB-IREP sometimes suffers from overfitting. As the figures for the breast-cancer and horse-colic datasets demonstrate, this may not only lead to overly large rule sets, it can also degrade accuracy.

5.4.2 Using significance testing

Significance tests can be used to detect whether the observed performance of a rule is due to chance alone, and are thus potentially useful in combating the overfitting problem observed with TB-IREP. As discussed above, there are two operations in the algorithm where it makes sense to integrate significance testing: rule selection and rule pruning. The following two sections evaluate these potential improvements empirically.

Significance testing for rule selection

TB-IREP's decision tree inducer returns several promising candidates for inclusion in the rule set. In the experiments reported in the last section, the most accurate rule on the pruning data is appended to the rules that have been generated so far. However, it is not clear whether this is the best choice because the rule's apparent edge in performance might just be due to sampling variation. This calls for significance testing. In the following experiments the most significant eligible leaf is included in the rule set instead of the most accurate one. As explained in Section 5.2.1, a leaf is eligible if it is more accurate than all its predecessors. Significance is measured by the p-value of the one-sided version of Fisher's exact test.

Table 5.3: Accuracy and number of rules for with (TB-IREP$_{sig}$) and without (TB-IREP) significance-based rule selection

	Accuracy			Number of rules		
Dataset	TB-IREP$_{sig}$	TB-IREP		TB-IREP$_{sig}$	TB-IREP	
anneal	98.2±0.4	98.3±0.4		11.5±0.5	13.8±0.4	•
audiology	76.2±1.4	76.5±1.1		15.8±0.4	20.0±0.3	•
australian	84.2±0.7	84.5±1.2		31.3±1.2	37.1±1.2	•
autos	72.3±2.0	72.0±2.2		18.7±0.8	23.1±0.8	•
balance-scale	81.5±1.1	81.4±1.0		30.6±0.7	36.1±1.0	•
breast-cancer	68.6±1.8	68.3±2.3		27.6±1.5	29.6±1.1	•
breast-w	95.0±0.5	94.8±0.6		9.6±0.7	11.8±0.9	•
german	69.8±1.2	71.0±0.7	○	61.0±1.6	71.3±2.4	•
glass-2	78.9±1.6	77.9±2.4		5.8±0.5	9.1±0.7	•
glass	68.4±2.5	69.0±2.5		12.0±0.6	15.6±0.6	•
heart-c	78.2±1.7	78.5±1.6		16.3±0.4	19.8±1.4	•
heart-h	80.6±1.6	80.3±1.6		15.5±0.9	18.5±0.8	•
heart-statlog	79.5±1.2	78.9±1.5		13.3±0.8	16.2±0.9	•
hepatitis	81.2±2.2	79.5±2.6		7.7±0.6	9.1±0.5	•
horse-colic	80.6±1.0	80.9±1.8		19.4±0.7	21.8±0.7	•
ionosphere	90.3±0.9	90.2±1.0		6.3±0.5	9.6±0.6	•
iris	94.9±0.6	94.9±0.9		3.7±0.2	4.0±0.2	•
labor	83.4±4.8	83.9±4.7		3.0±0.3	3.5±0.4	•
lymphography	78.7±2.3	76.0±2.3	•	10.0±0.5	11.2±0.4	•
pima-indians	73.7±1.6	72.8±1.2		10.2±0.8	21.8±1.8	•
primary-tumor	38.7±1.2	39.0±1.0		33.6±0.8	44.4±1.9	•
sonar	73.9±3.1	73.1±1.7		6.8±0.6	10.7±0.8	•
soybean	91.2±0.6	91.9±0.4	○	35.0±0.8	36.9±1.1	•
vehicle	71.8±1.1	72.6±1.0		26.6±1.3	42.5±3.0	•
vote	95.1±0.5	95.1±0.6		7.5±0.5	8.1±0.6	•
vowel	76.6±1.3	77.6±0.9		53.6±1.2	70.2±1.9	•
zoo	90.3±2.2	90.0±3.0		7.2±0.3	7.3±0.3	

Table 5.4: Results of paired t-tests (p=0.05): number indicates how often method in column significantly outperforms method in row

	Accuracy		Number of rules	
	TB-IREP$_{sig}$	TB-IREP	TB-IREP$_{sig}$	TB-IREP
TB-IREP$_{sig}$	–	2	–	0
TB-IREP	1	–	26	–

Performance is estimated in the same way as in the previous section. Table 5.3 displays the estimated number of rules and the rule sets' accuracy, and Table 5.4 summarizes the results of the paired two-sided t-tests. They show that significance testing reduces the number of rules for almost all of the 27 datasets. Only on the zoo dataset is the size of the rule set approximately the same. In some cases the decrease in size is quite dramatic. For the glass-2, ionosphere, pima-indians, vehicle, and sonar datasets, the number of rules is reduced by more than 30%, without any significant effect on accuracy. It is interesting that these five datasets are all purely numeric. A possible explanation for this is that the decision tree inducer generates many small leaves to approximate the non-linear class boundaries in these domains. With purely accuracy-based selection these leaves are often included in the beginning of the rule set instead of more general rules that cover a much larger portion of the instance space.

On almost all datasets significance-based rule selection has little influence on accuracy. It significantly increases accuracy on one dataset (lymphography), and significantly degrades it on two (german and soybean). This means that significance-based rule selection generally has a positive effect on the rule sets induced by TB-IREP. It substantially decreases their size with little effect on their accuracy.

Significance testing for pruning

Significance testing for rule selection reduces the number of rules that are generated, but it cannot prevent individual rules from overfitting the data. As discussed in Section 5.2.3, significance testing must be incorporated into the pruning process to accomplish effective overfitting avoidance for individual rules. With this modification, a rule is pruned if and only if none of its successors is both statistically significant and more accurate. Moreover, only those leaves are candidates for selection that are both sufficiently accurate as well as statistically significant.

The aggressiveness of the pruning process depends on the significance level that is employed. Recall that a rule is tested for significance with respect to all its prefixes. Only if all these significance tests are positive does the rule become a candidate for inclusion in the rule set. This means that even moderate significance levels are likely to incur substantial pruning. As in the case of post-pruning decisions trees

(Section 3.4), the following significance levels are used in the experiments for this section: 0.1, 0.2, 0.3, and 0.4. The p-value of the statistically eligible rules is used for rule selection.

Table 5.5 compares the accuracy of the rule sets for these significance levels to the accuracy of the rules generated by TB-IREP without significance testing for pruning—the latter corresponds to a significance level of 1.0. Table 5.6 summarizes the outcome of the paired two-tailed t-tests on these accuracy estimates. It shows that applying pruning at the 0.1-level significantly decreases accuracy on 15 datasets, significantly increasing it on only 3. Note that two of those three datasets (german and horse-colic) are the ones where IREP is more accurate than standard TB-IREP. Testing at the 0.2 and 0.3-levels changes this balance to 14 versus 3 and 10 versus 4 respectively. At the 0.4-level the gap in performance is relatively small. Accuracy significantly decreases on eight datasets, and significantly increases on four. The datasets where aggressive pruning has a beneficial or neutral effect are almost exclusively two-class datasets, those where it has a distinctly negative effect almost exclusively multi-class ones (the exceptions being the labor and sonar datasets). These results show that liberal significance testing is required if the main objective is to maximize accuracy.

However, sometimes comprehensibility is more important than maximum accuracy and in that case more aggressive pruning is appropriate. Table 5.7 compares the number of rules for the four different significance levels to the number of rules generated by applying no significance-based pruning, and Table 5.8 summarizes the outcome of the corresponding t-tests. The picture is clear: lowering the threshold reduces the number of rules on all 27 datasets significantly. As expected, the 0.1 significance level generates very small rule sets. It produces fewer than 5 rules on 16 of the datasets. Thus it provides an efficient means of summarizing the essential information in a dataset. However, even testing at the 0.4 level can reduce the number of rules dramatically. On 14 datasets it produces more than 30% fewer rules.

Minimizing the rule sets' size

The above experiments show that pruning is often harmful as far as the rule sets' accuracy is concerned. However, there are a few cases where it actually increases

Table 5.5: Accuracy for TB-IREP$_{sig}$ with different levels of significance-based pruning

	TB-IREP$_{sig}$ 1.0	TB-IREP$_{sig}$ 0.1	TB-IREP$_{sig}$ 0.2	TB-IREP$_{sig}$ 0.3	TB-IREP$_{sig}$ 0.4
anneal	98.2±0.4	97.2±0.4 •	97.7±0.2 •	97.9±0.3 •	98.0±0.3 •
audiology	76.2±1.4	69.3±2.5 •	72.0±1.7 •	72.9±1.8 •	74.3±1.9 •
australian	84.2±0.7	84.3±1.1	84.6±0.8	84.5±0.4	84.8±0.9 ◦
autos	72.3±2.0	62.9±2.1 •	65.2±3.1 •	69.3±2.1 •	69.8±2.6 •
balance-scale	81.5±1.1	78.6±0.9 •	80.3±1.4	81.3±1.2	81.4±1.1
breast-cancer	68.6±1.8	70.2±1.6	72.1±1.8 ◦	72.2±1.9 ◦	71.7±2.3 ◦
breast-w	95.0±0.5	94.2±0.5 •	94.7±0.4	94.8±0.6	95.0±0.6
german	69.8±1.2	71.2±0.9 ◦	71.7±0.9 ◦	72.1±0.7 ◦	71.6±1.1 ◦
glass-2	78.9±1.6	76.7±2.5 •	77.2±2.2	77.4±3.5	77.2±3.9
glass	68.4±2.5	63.7±3.7 •	64.0±3.0 •	65.4±2.5 •	67.5±2.2
heart-c	78.2±1.7	78.6±1.3	78.6±1.5	78.7±1.5	78.2±1.5
heart-h	80.6±1.6	79.1±1.9	78.8±0.9 •	79.5±1.4	79.4±1.3 •
heart-statlog	79.5±1.2	77.9±1.8	78.7±1.6	78.9±1.7	79.0±1.6
hepatitis	81.2±2.2	79.6±2.4	78.3±1.7 •	78.3±3.1 •	79.3±3.5
horse-colic	80.6±1.0	85.3±0.6 ◦	85.0±0.9 ◦	84.7±0.6 ◦	83.4±1.1 ◦
ionosphere	90.3±0.9	88.9±1.1 •	89.4±0.9 •	89.8±1.1	89.7±1.2 •
iris	94.9±0.6	94.8±1.1	95.3±0.9	95.2±0.8	95.2±0.8
labor	83.4±4.8	79.5±6.4 •	80.9±5.2 •	82.2±4.7	82.7±4.9
lymphography	78.7±2.3	75.0±2.1 •	75.6±3.0 •	76.1±2.1 •	76.4±2.7 •
pima-indians	73.7±1.6	73.3±1.0	73.7±1.1	73.6±0.9	73.8±1.2
primary-tumor	38.7±1.2	38.5±1.5	38.5±2.2	39.0±2.0	39.8±1.4
sonar	73.9±3.1	70.7±3.0 •	70.3±3.2 •	70.6±3.0 •	71.1±2.6 •
soybean	91.2±0.6	89.6±0.9 •	90.4±1.0 •	90.9±0.7	91.1±0.5
vehicle	71.8±1.1	69.5±1.0 •	69.8±0.8 •	70.3±1.0 •	70.8±1.2
vote	95.1±0.5	95.7±0.2 ◦	95.5±0.4	95.6±0.5 ◦	95.3±0.4
vowel	76.6±1.3	65.1±1.4 •	69.4±1.5 •	71.0±1.4 •	73.4±0.6 •
zoo	90.3±2.2	81.6±1.8 •	84.3±2.3 •	86.6±2.8 •	89.4±1.9

Table 5.6: Results of paired t-tests (p=0.05): number indicates how often method in column significantly outperforms method in row

	TB-IREP$_{sig}$ 1.0	TB-IREP$_{sig}$ 0.1	TB-IREP$_{sig}$ 0.2	TB-IREP$_{sig}$ 0.3	TB-IREP$_{sig}$ 0.4
TB-IREP$_{sig}$ 1.0	–	3	3	4	4
TB-IREP$_{sig}$ 0.1	15	–	9	11	13
TB-IREP$_{sig}$ 0.2	14	1	–	6	7
TB-IREP$_{sig}$ 0.3	10	1	0	–	4
TB-IREP$_{sig}$ 0.4	8	2	1	1	–

Table 5.7: Number of rules for TB-IREP$_{sig}$ with different levels of significance-based pruning

	TB-IREP$_{sig}$ 1.0	TB-IREP$_{sig}$ 0.1	TB-IREP$_{sig}$ 0.2	TB-IREP$_{sig}$ 0.3	TB-IREP$_{sig}$ 0.4
anneal	11.5±0.5	8.3±0.5 ◦	9.0±0.3 ◦	9.6±0.4 ◦	10.1±0.4 ◦
audiology	15.8±0.4	8.8±0.5 ◦	10.4±0.6 ◦	11.4±0.4 ◦	12.6±0.7 ◦
australian	31.3±1.2	4.2±0.3 ◦	6.1±0.5 ◦	8.4±0.9 ◦	12.0±0.4 ◦
autos	18.7±0.8	7.0±0.7 ◦	9.7±0.7 ◦	11.8±1.1 ◦	13.6±1.1 ◦
balance-scale	30.6±0.7	8.5±0.5 ◦	11.6±0.4 ◦	15.0±0.6 ◦	17.8±0.9 ◦
breast-cancer	27.6±1.5	2.5±0.3 ◦	4.5±0.5 ◦	6.5±0.8 ◦	10.9±1.3 ◦
breast-w	9.6±0.7	4.2±0.2 ◦	5.1±0.2 ◦	5.8±0.4 ◦	6.4±0.4 ◦
german	61.0±1.6	8.2±0.9 ◦	15.4±1.0 ◦	22.6±1.2 ◦	30.6±1.1 ◦
glass-2	5.8±0.5	2.5±0.3 ◦	3.0±0.2 ◦	3.4±0.2 ◦	3.9±0.2 ◦
glass	12.0±0.6	5.8±0.6 ◦	7.1±0.3 ◦	7.9±0.4 ◦	8.9±0.5 ◦
heart-c	16.3±0.4	3.8±0.2 ◦	5.0±0.4 ◦	6.1±0.6 ◦	7.8±0.7 ◦
heart-h	15.5±0.9	3.7±0.3 ◦	5.1±0.5 ◦	6.5±0.5 ◦	8.5±0.7 ◦
heart-statlog	13.3±0.8	3.5±0.3 ◦	4.5±0.3 ◦	5.5±0.4 ◦	6.9±0.6 ◦
hepatitis	7.7±0.6	2.0±0.1 ◦	2.5±0.2 ◦	3.3±0.2 ◦	3.9±0.4 ◦
horse-colic	19.4±0.7	2.9±0.3 ◦	3.9±0.3 ◦	6.0±0.5 ◦	8.4±0.7 ◦
ionosphere	6.3±0.5	3.0±0.3 ◦	3.4±0.3 ◦	3.8±0.5 ◦	4.3±0.6 ◦
iris	3.7±0.2	3.0±0.0 ◦	3.1±0.1 ◦	3.1±0.1 ◦	3.2±0.1 ◦
labor	3.0±0.3	1.9±0.1 ◦	2.1±0.1 ◦	2.3±0.2 ◦	2.4±0.2 ◦
lymphography	10.0±0.5	3.5±0.2 ◦	4.4±0.5 ◦	5.1±0.6 ◦	6.0±0.6 ◦
pima-indians	10.2±0.8	4.9±0.3 ◦	5.7±0.5 ◦	6.8±0.3 ◦	7.7±0.6 ◦
primary-tumor	33.6±0.8	7.4±0.8 ◦	11.0±0.7 ◦	14.5±0.7 ◦	18.7±1.2 ◦
sonar	6.8±0.6	2.9±0.3 ◦	3.6±0.2 ◦	4.2±0.4 ◦	4.8±0.3 ◦
soybean	35.0±0.8	24.8±0.8 ◦	26.7±0.6 ◦	28.0±0.8 ◦	29.3±0.5 ◦
vehicle	26.6±1.3	10.0±0.6 ◦	13.3±0.7 ◦	16.0±0.9 ◦	19.6±1.4 ◦
vote	7.5±0.5	3.0±0.3 ◦	3.3±0.3 ◦	3.8±0.3 ◦	4.3±0.2 ◦
vowel	53.6±1.2	26.0±0.8 ◦	33.0±1.1 ◦	37.6±0.9 ◦	42.8±1.1 ◦
zoo	7.2±0.3	4.7±0.1 ◦	5.4±0.2 ◦	5.8±0.2 ◦	6.5±0.3 ◦

Table 5.8: Results of paired t-tests (p=0.05): number indicates how often method in column significantly outperforms method in row

	TB-IREP$_{sig}$ 1.0	TB-IREP$_{sig}$ 0.1	TB-IREP$_{sig}$ 0.2	TB-IREP$_{sig}$ 0.3	TB-IREP$_{sig}$ 0.4
TB-IREP$_{sig}$ 1.0	–	27	27	27	27
TB-IREP$_{sig}$ 0.1	0	–	0	0	0
TB-IREP$_{sig}$ 0.2	0	26	–	0	0
TB-IREP$_{sig}$ 0.3	0	27	26	–	0
TB-IREP$_{sig}$ 0.4	0	27	27	27	–

Table 5.9: Accuracy and number of rules for TB-IREP$^{var}_{sig}$, compared to performance of TB-IREP$_{sig}$ without significance-based pruning

	Accuracy			Number of rules		
Dataset	TB-IREP$^{var}_{sig}$	TB-IREP$_{sig}$		TB-IREP$^{var}_{sig}$	TB-IREP$_{sig}$	
anneal	98.2±0.4	98.2±0.4		11.5±0.5	11.5±0.5	
audiology	76.2±1.4	76.2±1.4		15.8±0.4	15.8±0.4	
australian	84.7±1.2	84.2±0.7		3.6±0.2	31.3±1.2	•
autos	72.3±2.0	72.3±2.0		18.7±0.8	18.7±0.8	
balance-scale	81.5±1.1	81.5±1.1		30.6±0.7	30.6±0.7	
breast-cancer	72.2±1.9	68.6±1.8	•	6.5±0.8	27.6±1.5	•
breast-w	94.0±0.7	95.0±0.5	○	3.7±0.2	9.6±0.7	•
german	70.6±0.6	69.8±1.2		5.0±0.9	61.0±1.6	•
glass-2	77.4±3.5	78.9±1.6		3.4±0.2	5.8±0.5	•
glass	68.4±2.5	68.4±2.5		12.0±0.6	12.0±0.6	
heart-c	78.6±1.3	78.2±1.7		3.8±0.2	16.3±0.4	•
heart-h	79.7±1.4	80.6±1.6		6.3±0.7	15.5±0.9	•
heart-statlog	77.9±1.8	79.5±1.2		3.5±0.3	13.3±0.8	•
hepatitis	81.2±2.2	81.2±2.2		7.7±0.6	7.7±0.6	
horse-colic	85.3±0.6	80.6±1.0	•	2.9±0.3	19.4±0.7	•
ionosphere	88.9±1.1	90.3±0.9	○	3.0±0.3	6.3±0.5	•
iris	94.9±0.6	94.9±0.6		3.7±0.2	3.7±0.2	
labor	83.4±4.8	83.4±4.8		3.0±0.3	3.0±0.3	
lymphography	78.7±2.3	78.7±2.3		10.0±0.5	10.0±0.5	
pima-indians	72.9±0.9	73.7±1.6		4.4±0.3	10.2±0.8	•
primary-tumor	38.7±1.2	38.7±1.2		33.6±0.8	33.6±0.8	
sonar	70.6±3.0	73.9±3.1	○	4.2±0.4	6.8±0.6	•
soybean	91.2±0.6	91.2±0.6		35.0±0.8	35.0±0.8	
vehicle	69.5±1.0	71.8±1.1	○	10.0±0.6	26.6±1.3	•
vote	95.7±0.2	95.1±0.5	•	3.0±0.3	7.5±0.5	•
vowel	71.0±1.4	76.6±1.3	○	37.6±0.9	53.6±1.2	•
zoo	90.3±2.2	90.3±2.2		7.2±0.3	7.2±0.3	

Table 5.10: Results of paired t-tests (p=0.05): number indicates how often method in column significantly outperforms method in row

	Accuracy		Number of rules	
	TB-IREP$^{var}_{sig}$	TB-IREP$_{sig}$	TB-IREP$^{var}_{sig}$	TB-IREP$_{sig}$
TB-IREP$^{var}_{sig}$	–	5	–	0
TB-IREP$_{sig}$	3	–	15	–

Table 5.11: Thresholds for choosing the significance level for pruning in TB-IREP$_{sig}^{var}$

Size of smallest class	Significance level
$n_{min} < 50$	1.0
$50 \leq n_{min} < 100$	0.3
$100 \leq n_{min} < 200$	0.1
$200 \leq n_{min} < 400$	0.05
$400 \leq n_{min}$	0.01

accuracy, and on several datasets the size of the rule sets can be reduced significantly without affecting accuracy at all. The problem is to choose the right significance level. Recall that in Chapter 3 the right significance level for reduced-error pruning in decision trees is chosen via cross-validation. The same method can be applied here. However, the procedure is rather slow because the models for the different significance levels are not nested as in the case of decision trees. This means that a rule set must be built from scratch for each fold of the cross-validation and each significance level investigated.

Therefore we adopt a simpler heuristic that determines an appropriate significance level according to the amount of data available. As explained above, significance testing is unlikely to find a significant association when little data is present. This is a particular problem when there are many classes and some of the classes cover only very few instances—because aggressive significance testing is unlikely to generate a rule for a class that is populated sparsely. Thus it makes sense to choose the significance level according to the number of instances n_{min} in the smallest class. This is the strategy we adopt for the experiments in this section. The significance level is chosen according to the thresholds displayed in Table 5.11, in a similar manner to the procedure suggested by Martin (1997) in the context of pre-pruning decision trees. To derive these thresholds we started with the values employed by Martin and modified them to produce good results on our datasets.[3] Martin used the exact probability of a contingency table instead of Fisher's exact test and considered the total number of instances in a dataset instead of the number

[3]To be specific, we replaced 500 as the largest threshold on the number of instances with 400 because none of the datasets used in our experiments had 500 or more instances in its smallest class, and chose 0.3 instead of 0.5 as the second value for the significance level because 0.5 resulted in almost no pruning at all.

of instances in its smallest class.

Tables 5.9 and 5.10 summarize the performance of the resulting method, called TB-IREP$_{sig}^{var}$, comparing it to the performance of TB-IREP$_{sig}$, the same procedure without significance-based pruning. The results show that this simple strategy of adapting the significance level generally works well. On the 15 datasets where it significantly reduces the rule sets' size, it significantly increases accuracy in three cases, significantly decreasing it in five. It is interesting that all five of these datasets contain exclusively numeric attributes. Thus, in contrast to significance-based rule selection, which does not negatively affect accuracy on datasets of this type, significance-based pruning decreases performance in several cases. We conjecture that this is because additional pruning discards many of the rules in the tail of the rule set, and these rules may be useful in approximating non-linear class boundaries in numeric datasets.

5.4.3 Building partial trees

Partial trees are a way of avoiding building the full decision tree to obtain a single rule. This section investigates empirically whether the PART procedure has a negative impact on performance. It also contains experimental results comparing the time complexity of inducing a rule set with TB-IREP on the one hand, and PART on the other.

Error-based

This section compares the performance of the two algorithms when no significance testing is applied. Table 5.12 shows the estimated accuracies and rule sizes for both methods. Table 5.13 contains the number of significant wins and losses. As above, ten ten-fold cross-validation runs are used to estimate performance, and estimates are judged to be significantly different at the 5%-level according to a two-sided paired t-test.

These results indicate that building partial trees has almost no influence on the rule sets' accuracy. Although PART is significantly less accurate on two datasets (german and vehicle), the absolute difference in performance on these datasets is small. However, PART frequently produces larger rule sets than TB-IREP. On 19

Table 5.12: Accuracy and number of rules for TB-IREP compared to PART.

	Accuracy			Number of rules		
Dataset	PART	TB-IREP		PART	TB-IREP	
anneal	98.4±0.3	98.3±0.4		14.8±0.7	13.8±0.4	∘
audiology	76.3±1.5	76.5±1.1		22.3±0.7	20.0±0.3	∘
australian	84.0±1.2	84.5±1.2		51.2±2.2	37.1±1.2	∘
autos	72.0±3.1	72.0±2.2		24.3±1.2	23.1±0.8	∘
balance-scale	80.5±1.0	81.4±1.0		45.6±1.3	36.1±1.0	∘
breast-cancer	67.0±3.3	68.3±2.3		42.7±1.3	29.6±1.1	∘
breast-w	94.7±0.5	94.8±0.6		13.3±0.8	11.8±0.9	∘
german	70.0±1.0	71.0±0.7	∘	108.6±2.4	71.3±2.4	∘
glass-2	77.4±3.1	77.9±2.4		9.2±0.8	9.1±0.7	
glass	66.4±2.5	69.0±2.5		16.7±1.0	15.6±0.6	∘
heart-c	77.8±2.0	78.5±1.6		24.0±0.9	19.8±1.4	∘
heart-h	80.9±1.9	80.3±1.6		20.1±0.9	18.5±0.8	∘
heart-statlog	78.1±2.3	78.9±1.5		19.6±1.0	16.2±0.9	∘
hepatitis	80.2±2.9	79.5±2.6		9.4±0.6	9.1±0.5	
horse-colic	80.2±1.4	80.9±1.8		27.9±1.7	21.8±0.7	∘
ionosphere	90.3±1.1	90.2±1.0		9.4±1.0	9.6±0.6	
iris	95.0±0.7	94.9±0.9		4.1±0.2	4.0±0.2	∘
labor	84.2±4.9	83.9±4.7		3.5±0.5	3.5±0.4	
lymphography	76.2±1.5	76.0±2.3		12.6±0.7	11.2±0.4	∘
pima-indians	72.5±1.0	72.8±1.2		15.4±1.3	21.8±1.8	•
primary-tumor	39.3±1.5	39.0±1.0		44.3±0.6	44.4±1.9	
sonar	71.4±2.9	73.1±1.7		9.4±0.3	10.7±0.8	•
soybean	91.7±0.5	91.9±0.4		41.9±0.5	36.9±1.1	∘
vehicle	71.1±1.2	72.6±1.0	∘	48.6±2.3	42.5±3.0	∘
vote	95.1±0.7	95.1±0.6		10.1±0.8	8.1±0.6	∘
vowel	77.6±1.6	77.6±0.9		99.2±3.0	70.2±1.9	∘
zoo	91.1±1.3	90.0±3.0		7.5±0.3	7.3±0.3	

Table 5.13: Results of paired t-tests (p=0.05): number indicates how often method in column significantly outperforms method in row

	Accuracy		Number of rules	
	PART	TB-IREP	PART	TB-IREP
PART	–	2	–	19
TB-IREP	0	–	2	–

datasets it generates significantly more rules, and significantly less on only two. On the breast-cancer and german dataset PART outputs 30% more rules than TB-IREP. This is a direct consequence of the fact that PART's pruning operation is inherently less aggressive. On the other hand, it is surprising to see that PART produces smaller rule sets on two datasets (pima-indians and sonar). This must be attributed to the fact that PART only explores a fraction of the full decision tree, and is therefore less likely to find a rule that fits the pruning data particularly well just by chance. Hence, although PART's pruning method is less aggressive, it can sometimes generate fewer rules than TB-IREP.

Using significance testing for rule selection

We have compared the performance of TB-IREP and PART when no significance testing is applied. Here we investigate the effect of adding significance testing for rule selection on the two methods' relative performance. The experimental procedure is the same as in the last section. Table 5.14 shows the absolute performance figures for the two methods, TB-IREP$_{sig}$ and PART$_{sig}$, and Table 5.15 summarizes the results of the t-tests.

As far as accuracy is concerned, Table 5.14 shows that the two methods perform similarly on almost all of the 27 datasets if significance-based rule selection is performed. PART$_{sig}$ is significantly less accurate on heart-statlog, and significantly more accurate on primary-tumor. The results also show that TB-IREP$_{sig}$ produces smaller rule sets than PART$_{sig}$ on almost all of the datasets. In contrast to the situation between PART and TB-IREP without significance-based selection, PART$_{sig}$ never produces significantly smaller rule sets than TB-IREP$_{sig}$.

Minimizing the rule sets' size

As in TB-IREP, significance testing must be employed to prevent individual rules from overfitting the pruning data. Again, the correct significance level depends on the properties of the domain. If the value is too low, overpruning will occur. If it is too high, the rule set becomes unnecessarily complex. As before, a good heuristic is to choose the significance level according to the amount of data that is available—more specifically, the number of instances in the dataset's smallest class. Again, we

Table 5.14: Accuracy and number of rules for TB-IREP$_{sig}$ compared to PART$_{sig}$

	Accuracy			Number of rules		
Dataset	PART$_{sig}$	TB-IREP$_{sig}$		PART$_{sig}$	TB-IREP$_{sig}$	
anneal	98.3±0.3	98.2±0.4		14.0±0.4	11.5±0.5	◦
audiology	75.0±1.3	76.2±1.4		17.8±0.6	15.8±0.4	◦
australian	84.0±1.0	84.2±0.7		42.2±1.6	31.3±1.2	◦
autos	70.6±2.1	72.3±2.0		20.9±1.0	18.7±0.8	◦
balance-scale	81.5±1.2	81.5±1.1		39.2±1.0	30.6±0.7	◦
breast-cancer	68.9±1.3	68.6±1.8		39.5±1.6	27.6±1.5	◦
breast-w	95.2±0.5	95.0±0.5		10.1±0.3	9.6±0.7	
german	70.3±1.4	69.8±1.2		89.0±1.8	61.0±1.6	◦
glass-2	78.1±2.6	78.9±1.6		6.0±0.5	5.8±0.5	
glass	66.6±1.4	68.4±2.5		13.1±0.4	12.0±0.6	◦
heart-c	77.5±1.5	78.2±1.7		18.9±0.9	16.3±0.4	◦
heart-h	80.5±2.4	80.6±1.6		16.7±0.9	15.5±0.9	◦
heart-statlog	77.6±1.5	79.5±1.2	◦	15.4±0.8	13.3±0.8	◦
hepatitis	81.9±2.2	81.2±2.2		8.2±0.4	7.7±0.6	
horse-colic	80.2±1.7	80.6±1.0		23.4±1.5	19.4±0.7	◦
ionosphere	90.6±0.7	90.3±0.9		6.3±0.6	6.3±0.5	
iris	95.1±0.6	94.9±0.6		3.8±0.2	3.7±0.2	◦
labor	83.3±5.3	83.4±4.8		3.0±0.3	3.0±0.3	
lymphography	76.7±3.5	78.7±2.3		11.2±0.6	10.0±0.5	◦
pima-indians	73.7±1.3	73.7±1.6		11.1±0.7	10.2±0.8	◦
primary-tumor	40.4±1.6	38.7±1.2	•	41.8±0.9	33.6±0.8	◦
sonar	73.4±3.4	73.9±3.1		7.1±0.4	6.8±0.6	
soybean	91.1±0.9	91.2±0.6		39.2±0.9	35.0±0.8	◦
vehicle	71.5±1.5	71.8±1.1		32.7±1.6	26.6±1.3	◦
vote	95.2±0.5	95.1±0.5		9.4±0.8	7.5±0.5	◦
vowel	75.6±1.7	76.6±1.3		78.5±1.8	53.6±1.2	◦
zoo	90.4±1.3	90.3±2.2		7.3±0.3	7.2±0.3	

Table 5.15: Results of paired t-tests (p=0.05): number indicates how often method in column significantly outperforms method in row

	Accuracy		Number of rules	
	PART$_{sig}$	TB-IREP$_{sig}$	PART$_{sig}$	TB-IREP$_{sig}$
PART$_{sig}$	–	1	–	20
TB-IREP$_{sig}$	1	–	0	–

make this choice according to the thresholds in Table 5.11.

Table 5.16 and Table 5.17 compare the performance of the two corresponding procedures, TB-IREP$_{sig}^{var}$ and PART$_{sig}^{var}$, when the same thresholds are used to select the appropriate significance level in both methods. They show that their relative performance remains virtually unchanged: the two methods are similarly accurate but TB-IREP$_{sig}^{var}$ produces smaller rule sets on most datasets. However, the absolute difference in the size of the rule sets is usually small. On two datasets, primary-tumor and vowel, PART$_{sig}^{var}$'s rule sets are both larger and more accurate, indicating that TB-IREP$_{sig}^{var}$ overprunes on this data.

Timing experiments

Building partial trees instead of full ones should lead to a significant speed-up if little pruning is done. This section verifies this empirically on the 27 practical datasets by comparing the runtime of TB-IREP$_{sig}$ and PART$_{sig}$ both with and without significance-based pruning. It also investigates how TB-IREP$_{sig}$ and PART$_{sig}$ scale to large noisy datasets. As shown above, the bound on the time complexity is $O(n^2 \log^2 n)$ for TB-IREP with significance-based pruning and $O(n^2 \log n)$ for PART. However, these upper bounds assume that the number of rules scales linearly with the number of training instances, which is unlikely to be the case in practice. This section shows that the time complexity is much lower even on very noisy datasets if significance-based pruning is performed.

Table 5.18 displays the runtime in seconds for PART$_{sig}$ and TB-IREP$_{sig}$ on the 27 practical datasets—both without significance-based pruning and with pruning at the 0.1% level.[4] It also includes the relative performance, measured by the ratio of the values obtained from the two methods. Table 5.18 shows that PART$_{sig}$ is indeed faster than TB-IREP$_{sig}$ if pruning is solely error-based: on average, it takes approximately 25% less time to generate a rule set. However, on large datasets with little noise the difference can be much larger. The vehicle dataset is a good example. On it, PART$_{sig}$ is three times faster than TB-IREP$_{sig}$. Note than sometimes PART$_{sig}$ is slower than TB-IREP$_{sig}$ because it generates more rules and this outweighs the fact that it constructs partial trees. With significance-based pruning

[4]Both algorithms were implemented in Java using the same underlying data structures and executed using the same virtual machine.

Table 5.16: Accuracy and number of rules for TB-IREP$^{var}_{sig}$ compared to PART$^{var}_{sig}$.

	Accuracy			Number of rules		
Dataset	PART$^{var}_{sig}$	TB-IREP$^{var}_{sig}$		PART$^{var}_{sig}$	TB-IREP$^{var}_{sig}$	
anneal	98.3±0.3	98.2±0.4		14.0±0.4	11.5±0.5	○
audiology	75.0±1.3	76.2±1.4		17.8±0.6	15.8±0.4	○
australian	84.3±1.4	84.7±1.2	○	3.9±0.3	3.6±0.2	○
autos	70.6±2.1	72.3±2.0		20.9±1.0	18.7±0.8	○
balance-scale	81.5±1.2	81.5±1.1		39.2±1.0	30.6±0.7	○
breast-cancer	72.0±1.2	72.2±1.9		10.2±1.4	6.5±0.8	○
breast-w	93.8±0.9	94.0±0.7		4.1±0.3	3.7±0.2	○
german	70.9±0.7	70.6±0.6		6.6±0.8	5.0±0.9	○
glass-2	78.2±3.1	77.4±3.5		3.9±0.2	3.4±0.2	○
glass	66.6±1.4	68.4±2.5		13.1±0.4	12.0±0.6	○
heart-c	78.1±1.3	78.6±1.3		4.5±0.5	3.8±0.2	○
heart-h	79.4±1.4	79.7±1.4		7.4±0.7	6.3±0.7	○
heart-statlog	78.9±2.4	77.9±1.8		4.1±0.3	3.5±0.3	○
hepatitis	81.9±2.2	81.2±2.2		8.2±0.4	7.7±0.6	
horse-colic	85.2±0.5	85.3±0.6		3.6±0.3	2.9±0.3	○
ionosphere	89.0±1.0	88.9±1.1		3.0±0.2	3.0±0.3	
iris	95.1±0.6	94.9±0.6		3.8±0.2	3.7±0.2	○
labor	83.3±5.3	83.4±4.8		3.0±0.3	3.0±0.3	
lymphography	76.7±3.5	78.7±2.3		11.2±0.6	10.0±0.5	○
pima-indians	73.5±1.1	72.9±0.9		4.7±0.2	4.4±0.3	○
primary-tumor	40.4±1.6	38.7±1.2	•	41.8±0.9	33.6±0.8	○
sonar	70.1±2.4	70.6±3.0		4.3±0.2	4.2±0.4	
soybean	91.1±0.9	91.2±0.6		39.2±0.9	35.0±0.8	○
vehicle	69.2±1.8	69.5±1.0		12.7±0.7	10.0±0.6	○
vote	95.7±0.2	95.7±0.2		3.0±0.2	3.0±0.3	
vowel	72.8±0.9	71.0±1.4	•	56.5±1.5	37.6±0.9	○
zoo	90.4±1.3	90.3±2.2		7.3±0.3	7.2±0.3	

Table 5.17: Results of paired t-tests (p=0.05): number indicates how often method in column significantly outperforms method in row

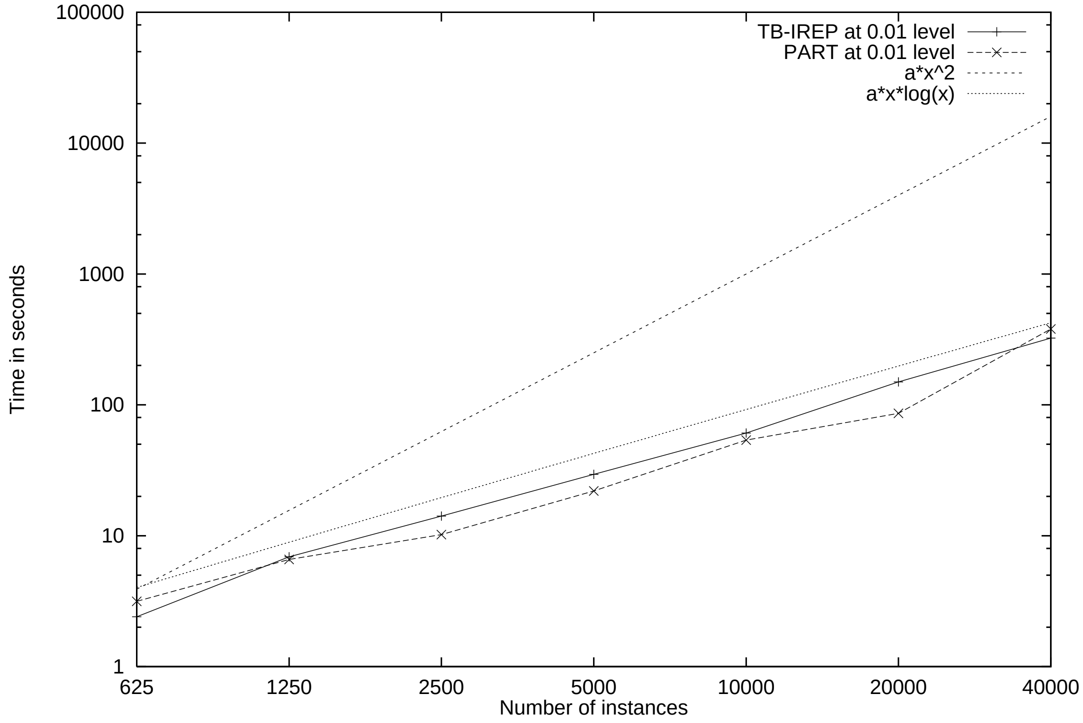
	Accuracy		Number of rules	
	PART$^{var}_{sig}$	TB-IREP$^{var}_{sig}$	PART$^{var}_{sig}$	TB-IREP$^{var}_{sig}$
PART$^{var}_{sig}$	–	1	–	21
TB-IREP$^{var}_{sig}$	2	–	0	–

Table 5.18: Induction times in seconds for PART_{sig} compared to TB-IREP_{sig}.

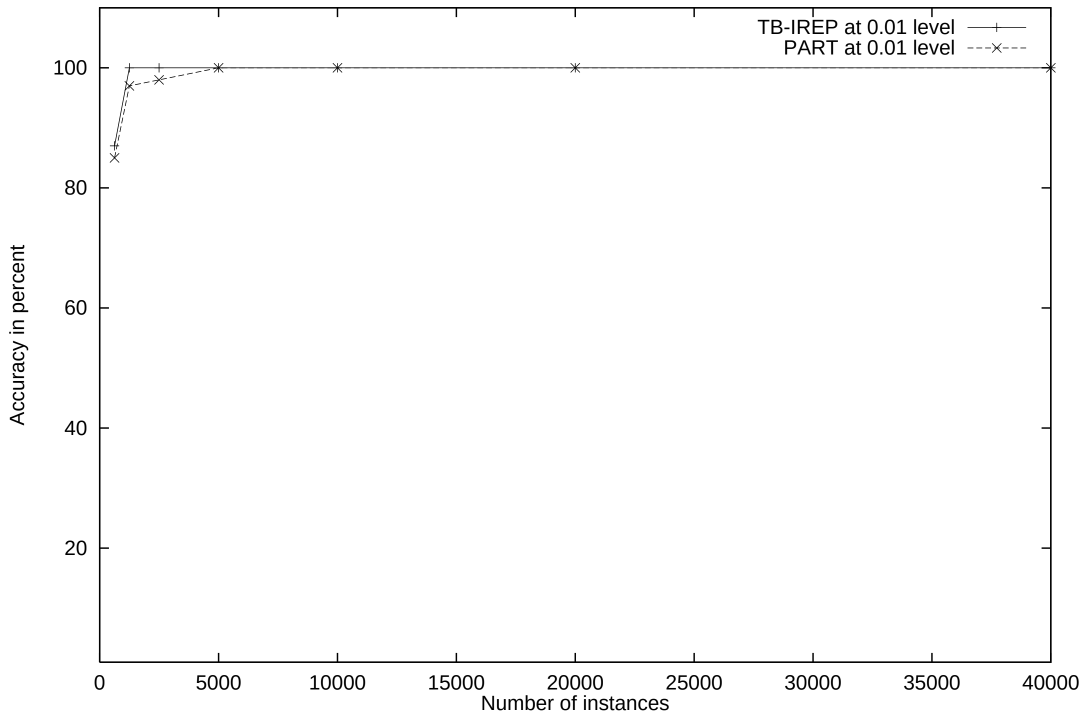
	No significance-based pruning			Pruning at 0.1 level		
Dataset	PART_{sig}	TB-IREP_{sig}	Ratio	PART_{sig}	TB-IREP_{sig}	Ratio
anneal	11.58	10.48	1.15	13.43	10.48	1.28
audiology	5.31	6.23	0.85	5.36	5.27	1.02
australian	7.93	12.44	0.64	3.13	3.47	0.90
autos	3.22	5.92	0.54	3.76	3.94	0.95
balance-scale	3.41	6.65	0.51	2.55	3.16	0.81
breast-cancer	2.41	3.09	0.78	1.16	1.21	0.96
breast-w	2.23	2.97	0.75	2.00	2.52	0.79
german	21.65	51.97	0.42	9.04	9.16	0.99
glass-2	1.01	1.05	0.96	0.93	0.90	1.03
glass	2.06	2.63	0.78	1.64	1.74	0.94
heart-c	2.33	2.39	0.97	1.39	1.57	0.89
heart-h	2.45	2.87	0.85	1.70	1.70	1.00
heart-statlog	2.47	4.69	0.53	1.88	1.70	1.16
hepatitis	1.20	1.21	0.99	0.89	0.92	0.97
horse-colic	4.37	4.70	0.93	2.46	2.11	1.17
ionosphere	6.84	6.85	1.00	4.38	4.31	1.02
iris	0.59	0.53	1.11	0.55	0.51	1.08
labor	0.56	0.61	0.92	0.51	0.44	1.16
lymphography	1.10	1.21	0.91	0.90	0.93	0.97
pima-indians	4.04	6.64	0.61	3.30	4.53	0.73
primary-tumor	4.83	14.23	0.34	3.53	4.34	0.81
sonar	5.07	5.78	0.88	4.30	5.04	0.85
soybean	10.31	20.44	0.50	10.51	18.59	0.56
vehicle	14.48	45.49	0.32	16.1	22.33	0.72
vote	1.76	1.84	0.96	1.44	1.50	0.96
vowel	47.37	94.65	0.50	35.93	53.12	0.68
zoo	0.66	0.70	0.94	0.66	0.66	1.00
average			0.76			0.94

at the 0.1 level—a setting that results in heavy pruning—the average runtime for PART_{sig} and TB-IREP_{sig} is similar. PART_{sig} is still slightly faster on average but the speed-up is rarely substantial.

Cohen (1995) presents a noisy artificial dataset on which the rule learner included in C4.5 (Quinlan, 1992) scales with the cube of the number of examples. He also shows that both IREP and RIPPER, an algorithm described in the same paper, exhibit a time complexity close to $O(n \log n)$ on this dataset. Figure 5.16a depicts the runtime for both TB-IREP_{sig} and PART_{sig} with significance-based pruning at the 0.01 level for the same dataset on a logarithmic scale. It shows that both algorithms also scale at about $O(n \log n)$ on this dataset. Although heavy pruning is



(a)



(b)

Figure 5.16: Runtime (a) and accuracy (b) of TB-IREP$_{sig}$ and PART$_{sig}$ on the artificial dataset ab+bcd+defg with 20% class noise, 12 irrelevant attributes, and uniformly distributed examples

performed, PART_{sig} is sometimes quicker than TB-IREP_{sig}— depending on whether it happens to choose the right subsets to expand. Figure 5.16b shows how well the generated rule sets approximate the target concept, represented by 100 noise-free test instances. It shows that both methods converge to the correct solution, with PART_{sig} taking four times longer than TB-IREP_{sig} to achieve 100% accuracy.

5.5 Related work

Separate-and-conquer rule learning has been an area of intense research in the machine learning community, and Fürnkranz (1999) presents a comprehensive review of work in this area. Incremental reduced-error pruning, the separate-and-conquer rule learning method most closely related to the algorithms presented in this chapter, has been discussed extensively above. Other work based on incremental reduced-error pruning and rule learning with decision trees is discussed in more detail below, along with methods for learning decision lists and other rule learners that apply significance testing in one form or another. However, there are other approaches to learning rules apart from separate-and-conquer algorithms and methods that convert a decision tree into a rule set.

One strand of research considers rule learning as a generalization of instance-based learning (Aha et al., 1991) where an instance can be expanded to a hyperrectangle (Salzberg, 1991; Wettschereck & Dietterich, 1995; Domingos, 1996). A hyperrectangle is the geometric representation of a rule in instance space. Whereas separate-and-conquer rule learning proceeds in a "top-down" fashion by splitting the instance space into small pieces, these methods operate "bottom-up" by clustering instances into hyperrectangles. During testing, an instance is classified using the "nearest" hyperrectangle in instance space. These distance-based methods have the disadvantage that they do not produce an explicit representation of all the induced knowledge because they rely on the distance metric to make predictions.

Another strand of research that is closely related to separate-and-conquer rule learning is boosting (Freund & Schapire, 1996; Schapire et al., 1997). Boosting produces a weighted combination of a series of so-called "weak" classifiers by iteratively re-weighting the training data according to the performance of the most recently generated weak learner. Predictions are made by weighting the classifications of the individual weak classifiers according to their error rate on the re-weighted data. Like boosting, separate-and-conquer rule learning uses weak learners and a re-weighting scheme: each rule represents a weak classifier and all instances have discrete weights, one or zero. They receive a weight of one if they are not covered by any of the rules learned so far and a weight of zero otherwise. Using a boosting algorithm for weak learners that can abstain from making a prediction when an instance is not covered by a rule (Schapire & Singer, 1998), it is possible to learn very accurate rule sets for two-class problems (Cohen & Singer, 1999). However, the boosted classifier does not make all the knowledge explicit because a substantial part of its predictive power is hidden in the weights derived by the boosting algorithm. Therefore the rules can no longer be interpreted as individual "nuggets" of knowledge. Note that the TB-IREP and PART algorithms for learning a single rule could be used as weak learners in the boosting framework, with the possible benefit of further performance improvements.

5.5.1 RIPPER

Like TB-IREP and PART, the rule learner RIPPER (Cohen, 1995) builds on the basic IREP algorithm. However, unlike the methods presented in this chapter, it does not address the overpruning problems at the core of IREP. It improves on the basic IREP algorithm by using a different stopping criterion and a post-processing step in which the rule set is adjusted to globally optimize performance. To deal with multi-class problems, RIPPER employs the same strategy as IREP: it learns a rule set for each class in turn and concatenates the results. This means that RIPPER does not learn a proper decision list where the rules for each class can appear anywhere in the rule set. Consequently the order in which the rules are output does not usually correlate with their importance. This is a potential drawback for two reasons. First, it makes it harder for the user to judge the relative importance of each rule. Second, it is advantageous to list the most important rules at the beginning of the rule set because rules are harder to interpret the further down the list they are. If preceding rules belong to a different class, their effect must be taken into account when a rule is interpreted.

Table 5.19: Accuracy and number of rules for PART_{sig}^{var} compared to RIPPER.

	Accuracy			Number of rules		
Dataset	PART_{sig}^{var}	RIPPER		PART_{sig}^{var}	RIPPER	
anneal	98.3±0.3	98.3±0.1		14.0±0.4	8.7±0.2	○
audiology	75.0±1.3	72.3±2.2	•	17.8±0.6	17.0±0.5	○
australian	84.3±1.4	85.3±0.7		3.9±0.3	4.1±0.5	
autos	70.6±2.1	72.0±2.0		20.9±1.0	12.4±0.5	○
balance-scale	81.5±1.2	81.0±1.1		39.2±1.0	11.3±0.5	○
breast-cancer	72.0±1.2	71.8±1.6		10.2±1.4	3.2±0.6	○
breast-w	93.8±0.9	95.6±0.6	○	4.1±0.3	5.6±0.3	•
german	70.9±0.7	71.4±0.7		6.6±0.8	4.1±0.5	○
glass-2	78.2±3.1	80.9±1.4	○	3.9±0.2	4.4±0.4	•
glass	66.6±1.4	66.7±2.1		13.1±0.4	8.1±0.4	○
heart-c	78.1±1.3	78.5±1.9		4.5±0.5	4.3±0.2	
heart-h	79.4±1.4	78.7±1.3		7.4±0.7	3.3±0.2	○
heart-statlog	78.9±2.4	79.0±1.4		4.1±0.3	4.0±0.2	
hepatitis	81.9±2.2	77.2±2.0	•	8.2±0.4	2.9±0.1	○
horse-colic	85.2±0.5	85.0±0.8		3.6±0.3	3.8±0.4	
ionosphere	89.0±1.0	89.2±0.8		3.0±0.2	5.0±0.5	•
iris	95.1±0.6	94.4±1.7		3.8±0.2	3.5±0.1	○
labor	83.3±5.3	83.5±3.9		3.0±0.3	3.3±0.1	•
lymphography	76.7±3.5	76.1±2.4		11.2±0.6	6.2±0.2	○
pima-indians	73.5±1.1	75.2±1.1	○	4.7±0.2	3.8±0.4	○
primary-tumor	40.4±1.6	38.5±0.8	•	41.8±0.9	8.9±0.6	○
sonar	70.1±2.4	75.7±1.9	○	4.3±0.2	4.7±0.3	•
soybean	91.1±0.9	92.1±0.5	○	39.2±0.9	27.4±0.8	○
vehicle	69.2±1.8	69.0±0.6		12.7±0.7	14.5±0.5	•
vote	95.7±0.2	95.6±0.3		3.0±0.2	2.6±0.2	○
vowel	72.8±0.9	69.6±1.9	•	56.5±1.5	41.5±1.0	○
zoo	90.4±1.3	87.8±2.4	•	7.3±0.3	7.0±0.2	

Table 5.20: Results of paired t-tests (p=0.05): number indicates how often method in column significantly outperforms method in row

	Accuracy		Number of rules	
	PART_{sig}^{var}	RIPPER	PART_{sig}^{var}	RIPPER
PART_{sig}^{var}	–	5	–	16
RIPPER	5	–	6	–

RIPPER replaces IREP's error-based stopping criterion with a criterion derived from the minimum description length principle. It stops adding rules to a rule set when the description length of the current rule set is more than 64 bits larger than the smallest description length observed so far. This heuristic is designed to overcome IREP's problem with early stopping and seems to work well in practice. It prevents the algorithm from stopping too early due to an erroneous rule—found because the algorithm chose to explore the wrong part of the instance space. Note that the tree-based pruning methods in this chapter are a more principled way of addressing the same problem.

Once a rule set has been generated, RIPPER greedily deletes rules so long as this decreases the rule set's overall description length. It follows this up with a post-processing step that revises the rule set to more closely approximate what would have been obtained by a more expensive global pruning strategy. To do this, it considers "replacing" or "revising" individual rules, guided by the error of the modified rule set on the pruning data. It then decides whether to leave the original rule alone or substitute its replacement or revision, a decision that is made according to the minimum description length heuristic. Note that the training data is split into new growing and pruning sets for each individual rule that is considered for replacement or revision. Consequently, when a rule is revised, it has access to some instances in the growing set that have previously been used for pruning it and vice versa. This is clearly a departure the basic idea of reduced-error pruning. However, it is likely to help combat IREP's overpruning problems.

Table 5.19 and Table 5.20 compare the accuracy and size of the rule sets generated by RIPPER with the decision lists produced by PART$_{sig}^{var}$.[5] They show that the performance of the two methods is similar with respect to accuracy. PART$_{sig}^{var}$ is significantly more accurate on five datasets, and significantly less accurate on five. However, RIPPER generally produces smaller rule sets. On sixteen datasets it generates significantly fewer rules than PART$_{sig}^{var}$, and significantly more on six. On most two-class datasets RIPPER's tendency to prune very aggressively is not harmful. However, on several multi-class datasets it degrades performance significantly

[5]Note that the attribute information for the datasets had to be modified for this experiment because RIPPER's MDL criterion is very sensitive to the number of attribute and class values. If an attribute value does not actually occur in the data, RIPPER's performance can deteriorate significantly.

(audiology, primary-tumor, vowel, zoo). PART$^{var}_{sig}$, on the other hand, has problems on purely numeric two-class datasets. On three of the five datasets (breast-w, glass-2, sonar) where it performs significantly worse than RIPPER, it also produces smaller rule sets, indicating a tendency to overprune. This means that the heuristic described in Section 5.4.2 chooses a significance level that is too conservative on these datasets.

5.5.2 C4.5

The rule learner in C4.5 (Quinlan, 1992) does not employ a separate-and-conquer method to generate a set of rules—it achieves this by simplifying an unpruned decision tree using a global optimization procedure (Quinlan, 1987b). It starts by growing an unpruned decision tree using the decision tree inducer included in the C4.5 software. Then it transforms each leaf of the decision tree into a rule. This initial rule set will usually be very large because no pruning has been done. Therefore C4.5 proceeds to prune it using various heuristics.

First, each rule is simplified separately by greedily deleting conditions in order to minimize the rule's estimated error rate. Following that, the rules for each class in turn are considered and a "good" subset is sought, guided by a criterion based on the minimum description length principle (Rissanen, 1978) (this is performed greedily, replacing an earlier method that used simulated annealing). The next step ranks the subsets for the different classes with respect to each other to avoid conflicts, and determines a default class. Finally, rules are greedily deleted from the whole rule set one by one, so long as this decreases the rule set's error on the training data. Like RIPPER, C4.5 does not generate a proper decision list—although the rules must be evaluated in order—because rules belonging to the same class are grouped together.

Unfortunately, the global optimization process is rather lengthy and time-consuming. Five separate stages are required to produce the final rule set. Cohen (1995) shows that C4.5 can scale with the cube of the number of examples on noisy datasets, for example, the artificial dataset used in Section 5.4.3 of this chapter. Note that Zheng (1998) finds that C4.5's runtime can be improved on some datasets by using a pruned decision tree instead of an unpruned one to obtain an initial set of rules. However, he does not show that this improves performance on the noisy datasets used by Cohen.

The experimental results in Tables C.1, C.2, C.3, and C.4 of Appendix C show that both PART$^{var}_{sig}$ and RIPPER produce smaller rule sets than C4.5 on the practical datasets used in this chapter; however, they are often less accurate. PART$^{var}_{sig}$, for example, produces significantly smaller rule sets on 22 datasets, and significantly larger ones on only three. It is significantly less accurate than C4.5 on 11 datasets, and significantly more accurate on only four. Note that C4.5 does not use the basic hold-out strategy of reduced-error pruning to generate a rule set. Instead it uses *all* the data for both growing the initial tree and pruning the rule set. It is plausible that this is the reason why it produces more accurate rules on the datasets used in these experiments.

To test this hypothesis, we implemented the basic PART method using the error-based pruning strategy from the decision tree learner in C4.5 (Frank & Witten, 1998a). This means that the partial decisions trees in PART are built from all the available training data and no pruning data is set aside. Once a partial tree has been generated, we select the leaf with maximum coverage for inclusion in the rule set—in other words, coverage is used as an approximation to the statistical significance of a rule.

Table 5.21 and Table 5.22 compare this modified rule learner, called PART$_{C4}$, to the rule inducer included in C4.5. The results show that PART$_{C4}$ and C4.5 are similarly accurate and produce rule sets of similar size. PART$_{C4}$ is significantly more accurate on seven datasets, and significantly less accurate on seven. It produces smaller rule sets on thirteen datasets, and larger ones on eleven. Note, however, that PART$_{C4}$ does not suffer from C4.5's problem of prohibitive runtime on noisy datasets—its worst-case time complexity is the same as that of PART with reduced-error pruning. Moreover, unlike C4.5, it generates a proper decision list, in which the rules are listed in order of their importance.

Tables C.5 and C.6 of Appendix C compare the performance of PART$_{C4}$ and RIPPER. They show that PART$_{C4}$ is significantly more accurate on eleven datasets and significantly less accurate on only six. However, RIPPER produces significantly fewer rules than PART$_{C4}$ on all 27 datasets.

Table 5.21: Accuracy and number of rules for PART_{C4} compared to C4.5

	Accuracy		Number of rules	
Dataset	PART_{C4}	C4.5	PART_{C4}	C4.5
anneal	98.4±0.3	98.6±0.2 ∘	14.7±0.3	17.2±0.4 •
audiology	78.7±1.0	76.3±1.0 •	19.6±0.3	21.4±0.3 •
australian	84.3±1.2	84.9±1.1	30.7±1.5	14.4±0.8 ∘
autos	74.5±1.1	76.7±2.6 ∘	20.5±0.8	24.8±0.8 •
balance-scale	82.3±1.2	77.9±0.6 •	38.6±1.0	37.1±0.8 ∘
breast-cancer	69.6±1.6	70.4±1.6	19.1±1.2	10.0±0.6 ∘
breast-w	94.9±0.4	95.5±0.6 ∘	10.0±0.5	10.1±0.4
german	70.0±1.4	71.5±0.7 ∘	69.6±1.3	27.2±1.5 ∘
glass-2	80.0±4.0	79.6±2.2	6.7±0.4	9.6±0.6 •
glass	69.8±2.3	66.6±3.0 •	15.3±0.6	14.6±0.6 ∘
heart-c	78.5±1.7	80.2±1.4	19.7±0.6	14.2±0.7 ∘
heart-h	80.5±1.5	79.5±1.4	8.8±0.6	8.3±0.5
heart-statlog	78.9±1.3	81.1±1.4 ∘	17.8±1.0	12.0±0.8 ∘
hepatitis	80.2±1.9	79.6±0.9	8.4±0.5	9.2±0.4 •
horse-colic	84.4±0.8	83.0±0.6 •	9.5±0.8	7.0±0.3 ∘
ionosphere	90.6±1.3	90.5±1.7	7.5±0.5	10.6±0.5 •
iris	93.7±1.6	95.0±1.0 ∘	4.0±0.4	5.0±0.1 •
labor	77.3±3.9	81.4±2.6 ∘	3.6±0.2	5.2±0.3 •
lymphography	76.5±2.7	77.7±1.9	11.6±0.5	11.3±0.3
pima-indians	73.6±0.5	74.2±1.1	7.4±0.4	11.3±0.8 •
primary-tumor	41.7±1.3	40.6±1.7 •	40.8±1.0	18.3±0.7 ∘
sonar	76.5±2.3	74.5±3.2 •	7.5±0.4	9.6±0.3 •
soybean	91.4±0.5	91.8±0.7	37.0±0.8	35.9±0.5 ∘
vehicle	72.4±0.8	72.8±1.0	33.3±1.2	29.0±2.3 ∘
vote	95.9±0.6	95.8±0.6	6.2±0.4	7.0±0.2 •
vowel	78.1±1.1	78.1±1.2	65.9±1.5	73.9±1.8 •
zoo	92.2±1.2	91.6±1.2 •	7.6±0.1	9.1±0.1 •

Table 5.22: Results of paired t-tests (p=0.05): number indicates how often method in column significantly outperforms method in row

	Accuracy		Number of rules	
	PART_{C4}	C4.5	PART_{C4}	C4.5
PART_{C4}	–	7	–	11
C4.5	7	–	13	–

5.5.3 Inducing decision lists

Decision lists were formally introduced by Rivest (1987). He shows that decision lists with at most k conjunctions in each rule are "polynomially learnable" in the sense defined by Valiant (1984). This notion of learnability assumes that the target concept can be described with 100% accuracy as a decision list with k conjunctions. If this is the case, Rivest's greedy decision list inducer outputs a hypothesis that is approximately correct with high probability. Moreover, its runtime is polynomial in the number of training instances. Rivest's algorithm uses the simple separate-and-conquer scheme from Figure 5.3, greedily adding conditions to a rule until k is exceeded or the rule is pure. However, the algorithm is not useful in practice because it assumes that the data is noise-free and that the target concept can be identified correctly using a k-term decision list.

Clark and Niblett (1989) present a decision list inducer called CN2 that includes a simple facility to deal with noisy data. Like Rivest's inducer, it uses the basic separate-and-conquer method to generate a rule set. However, instead of applying a greedy mechanism for generating a rule, CN2 uses beam search with a user-defined beam width k, starting the search with the empty rule. In each step of the beam search, it adds all possible specializations of the rules currently in the beam. Numeric attributes must be pre-discretized. The best k of these new rules make up the beam for the next iteration of the search, where rule quality is measured by the entropy of the rule's class distribution. Then, of all statistically significant rules encountered during the search, the one with the lowest entropy is appended to the rule set. Statistical significance is measured on the training data using the parametric chi-squared test based on the log-likelihood ratio statistic (Agresti, 1990), comparing the class distribution of the rule being tested to the class distribution for the default rule.

This is similar to the significance-based rule selection in TB-IREP and PART. However, the parametric chi-squared test based on the log-likelihood ratio statistic is known to be overly liberal with small and skewed samples, whereas Fisher's exact test does not suffer from this drawback (Agresti, 1990). A bigger problem is that the significance test is applied to the training data, to which the rule has been fitted. This dramatically increases the likelihood of finding a "significant" rule that is not really significant. Also, CN2 does not check whether a rule is a significant extension of a more general rule, as TB-IREP and PART do. In a later paper, Clark and Boswell (1991) observe that CN2 fails to find general rules, and replace the entropy-based evaluation criterion by a Laplace-adjusted error estimate (also obtained from the training data). However, their results show that the modified decision list inducer—without significance testing—is less accurate than an early version of C4.5 (Quinlan, 1987a) on nine of twelve benchmark datasets used in their study. With significance testing at the 0.05% level, it is less accurate than C4.5 on all datasets.

Pagallo and Haussler (1990) investigate the problem of subtree replication in decision trees. As a consequence, they propose a greedy algorithm called GROVE for learning a general decision list. GROVE uses the basic separate-and-conquer scheme from Figure 5.3 to generate an unpruned decision list. Then it employs global reduced-error pruning to greedily prune this list. GROVE's strategy for growing a single rule is closely related to the decision-tree-based approach in PART. Like PART, GROVE adds a test to a rule by first choosing the attribute that is most predictive, and then expanding the value with minimum entropy. PART can be viewed as an extension to GROVE that implements tree-based incremental pruning. Indurkhya and Weiss (1991) propose an extension to GROVE that, like PART, builds a decision tree to obtain a set of candidate rules. They present promising results on selected artificial datasets. However, their method prunes a rule *after* it has been extracted from the decision tree and consequently does not solve the problems associated with hasty generalization.

Usually decision lists are constructed by appending rules to an existing rule set. However, this is not the only way a decision list can be generated. Webb (1994) proposes an algorithm that produces a decision list by incrementally *prepending* rules to a rule set, starting with the default rule. The idea is to make maximum use of the default rule in order to produce small rule sets. However, Webb finds that this approach tends to sweep erroneous small disjuncts to the beginning of the rule set, giving them too much influence on future predictions, and so he adopts an approach that eliminates rules with too small a coverage. Instead of automatically prepending a rule to the beginning of the rule set, this new approach inserts new rules as deeply as possible in the list without decreasing overall accuracy. With this technique, Webb finds that prepending rules outperforms appending rules both with respect to accuracy and size of the resulting rule sets. Note, however, that no pruning is performed in his experiments. Also, prepending rules destroys the main advantage of using decision lists for the purpose of knowledge acquisition because the most important information can no longer be found at the beginning of the rule set.

Nock and Jappy (1998, 1999) formally show that it is NP-hard to find the smallest decision list consistent with a set of examples. Unless $P = NP$, which is not likely to be the case, this result means that there exists no algorithm with polynomial time complexity that generates the smallest decision list consistent with a set of examples. They also present a greedy algorithm for learning a decision list, similar to GROVE (Pagallo & Haussler, 1990). The algorithm, called IDCL, modifies a rule by adding the attribute-value test that minimizes the Gini index (Breiman et al., 1984) for both the current rule and the remaining instances. The Gini index is a splitting criterion similar to the information gain used by GROVE. After a fully expanded decision list has been grown, it is greedily pruned using global optimization on a hold-out set. However, unlike GROVE, which prunes by minimizing the error on the hold-out set, IDCL uses a heuristic that includes a penalty based on the rule set's size.

5.5.4 Significance testing in rule learners

In addition to the parametric test used in CN2 (Clark & Niblett, 1989), permutation tests have been proposed as a means of making statistically reliable decisions in rule learners. Gaines (1989) appears to be the first to propose the use of a permutation test in an induction algorithm. In his rule learner INDUCT, the p-value of Fisher's exact test, approximated using the Binomial distribution, is used to judge the quality of a rule and to control the process of growing a rule. However, his use of the p-value is problematic for two reasons. First, the p-value is obtained from the data that has been used to fit the rule. Gaines proposes to use the Bonferroni correction to adjust for this by computing the total number of rules in the instance space and adapting the p-value correspondingly. However, the total number of possible rules is usually gigantic, leading to very conservative p-values. The second problem is that pruning decisions are made by comparing the p-values of a rule and its extension. A rule may be extremely significant just because it covers a huge number of instances, whereas its extension is very accurate but slightly less significant because it has lower coverage. Thus making pruning decisions by comparing p-values leads to overpruning.

Jensen (1992) uses a conditional permutation test based on the chi-squared statistic in a prototypical rule learner that generates a *k-of-n* rule for two-class problems. In this type of rule, an instance is assigned to the positive class if k or more of the rule's total of n conditions are true; otherwise, it is assigned to the negative class. The permutation test is used to decide when to stop the greedy process of modifying or expanding the current rule. A 0.1% significance level is employed as the stopping criterion. As in PART and TB-IREP, the p-value is computed from a hold-out set. In experiments on two standard datasets, the resulting rule learner achieves accuracies comparable to those obtained from the decision tree inducer C4.5 (Quinlan, 1992).

5.6 Conclusions

This chapter investigates overpruning problems in incremental pruning procedures for separate-and-conquer rule learning algorithms, and identifies hasty generalization and early stopping as the main culprits for these problems. We present a new incremental pruning algorithm for decision lists that is based on the repeated generation of pruned decision trees. This new pruning algorithm—tree-based incremental reduced-error pruning—is designed to overcome the overpruning phenomenon in the standard incremental approach. It avoids hasty generalization because all the implications of a rule are explored before a pruning decision is made. It prevents early stopping because it eliminates the need for a stopping criterion: rule generation automatically terminates when the tree structure is pruned back to its root node, implying that the remaining data does not contain any additional information.

The first hypothesis of this chapter is that tree-based incremental reduced-error pruning produces more accurate rule sets than standard incremental reduced-error pruning. The experimental results on the benchmark datasets show that the tree-based approach indeed generates more accurate rule sets on most of these datasets. However, the resulting classifiers are often unnecessarily large. This indicates that the purely error-based version of this new pruning algorithm suffers from underpruning.

Underpruning is caused by the fact that a purely error-based criterion is sensitive to statistical fluctuations in the data. Significance tests are an obvious way of combating this problem. In this chapter we focus on one of them, namely a simple permutation test known as Fisher's exact test, and investigate whether it can be used to reduce the rule sets' size and increase their accuracy. The application of this test is twofold. First, it enables rules to be selected according to their statistical significance. This prevents fluke rules that survive the pruning process from being included in the rule set instead of potentially more predictive candidates. Assuming that the significance of a rule correlates well with its importance, significance-based selection has the further advantage that rules are listed in order of their importance in the resulting rule set. The second application of significance testing is pruning, where it can be used to eliminate those extensions of a rule that are not statistically significant.

The second hypothesis of this chapter is that significance testing makes the rule sets generated by tree-based incremental pruning both smaller and more accurate. The experimental results show that their size can indeed be reduced substantially if the statistical significance of a rule is used to decide which candidate rule to append to a decision list. However, accuracy remains largely unaffected by this modification. The results also show that often the rule sets' size can be further reduced by using significance testing to decide when to discard extensions of a rule. To maximize accuracy, the appropriate significance level for pruning must be chosen according to the properties of the domain. We present a simple heuristic for making this choice according to the size of the sample for each class. This heuristic results in a good compromise between accuracy and size of the resulting rule sets.

This chapter also describes a modified version of the tree-based pruning algorithm that generates partial trees instead of full ones. The modification is a direct result of a slightly more conservative pruning operation and leads to a substantial speed-up on practical datasets when little pruning is required. We show that both significance-based rule selection and pruning can be adapted in a straightforward way to work in conjunction with the modified algorithm.

The third hypothesis of this chapter is that rule pruning with partial trees generates rule sets that are as accurate and as small as those produced by using fully expanded trees. The experimental results show that the generated rules are indeed similarly accurate. However, they also show that rule pruning with partial trees increases the rule sets' size. This is a direct consequence of the fact that this pruning mechanism is inherently less aggressive.

The resulting algorithm, $\mathrm{PART}_{sig}^{var}$, compares favorably to the state-of-the-art rule learner RIPPER. Like $\mathrm{PART}_{sig}^{var}$, RIPPER is based on reduced-error pruning, but it requires a complex global optimization step to produce accurate rule sets. Its rule sets are often smaller but suffer from the potential drawback that they do not represent proper decision lists where the rules are ordered according to their importance. On the benchmark datasets used in this thesis, both RIPPER and $\mathrm{PART}_{sig}^{var}$ produce rule sets that are less accurate but smaller than those generated by the rule learner C4.5. However, PART_{C4}, which performs rule pruning with partial trees using the pruning mechanism from the decision tree inducer in C4.5, generates rule sets that are as accurate and small as those produced by C4.5. This indicates that C4.5's error-based pruning procedure produces more accurate classifiers than reduced-error pruning on the datasets used in this thesis—most likely due to the fact that the datasets are relatively small and C4.5's pruning strategy does not require data to be held out for pruning.

Chapter 6

Conclusions

Pruning mechanisms are an important component of practical learning algorithms for decision trees and lists and are essential for learning comprehensible and accurate classifiers in the presence of noise. This thesis shows that the concept of statistical significance plays a key role in making pruning decisions, and demonstrates how pruning algorithms can be improved by applying significance tests to identify superfluous components of a classifier. The main claim (Section 1.3) is:

Significance tests are useful tools for pruning decision trees and lists.

The main findings are summarized in this chapter. Section 6.1 brings together the results from the individual topics that are covered in this thesis, Section 6.2 identifies the most important contributions to research, and Section 6.3 points out some possibilities for future work.

6.1 Results

It is well known that overfitting makes pruning a requirement in practical learning algorithms. Overfitting occurs because (a) the process of obtaining the training set is subject to sampling variance, (b) the complexity of a classifier means that some of its components enjoy little support from the data, and (c) intense search can cause the learning algorithm to describe spurious patterns in the sample. To clarify the effect of sampling variance and complexity, we derived a simple quantitative relationship between the size of a disjunct and its expected error on future data, showing that sampling variance causes the error of a disjunct to increase when its size decreases, assuming that no search is involved in generating the classifier and given a constant probability of observing each class. Then we presented a simple experiment that illustrates how search intensifies the problem of error-prone disjuncts by comparing two classifiers of the same complexity: the first one selected randomly from the space of possible classifiers, and the second one generated by searching for a classifier with minimum error on the training data. In the context of these results we explained why significance tests are an appropriate tool for identifying and pruning error-prone components of classification models.

The usefulness of significance testing was investigated in three areas: post-pruning decision trees, pre-pruning decision trees, and pruning decision lists. In the case of post-pruning, we demonstrated that the performance of standard reduced-error pruning can be improved using significance testing if the significance level is chosen appropriately for each individual learning task: the resulting decision trees are just as accurate but often significantly smaller. The significance level must be adjusted to each dataset because (a) multiple tests are performed in the process of pruning and (b) the expected accuracy of the pruned classifier depends on the test's significance level (determining the frequency of Type I errors) as well as its statistical power (determining the frequency of Type II errors). The power associated with a particular significance level depends on the properties of the dataset. We show that an appropriate significance level can be determined automatically using cross-validation. We also compare the parametric chi-squared test to several variants of non-parametric permutation tests and show that they lead to almost identical results. Apparent differences can be eliminated by adjusting the significance level.

In another set of experiments we explored ways of using significance tests for pre-pruning, comparing it to the standard procedure that is based on the parametric chi-squared test. As with post-pruning, it was shown that there is virtually no difference in overall performance when permutation tests are used instead of the parametric test: obtaining best results is a matter of adjusting the significance level suitably. In both cases the optimum significance level depends on the domain. We have also evaluated whether the standard parametric procedure can be improved by locally adjusting for significance tests on multiple attributes. The results indicate that this is not the case and that the same performance can be achieved by adjusting the significance level globally. In addition to a comparison of different pre-pruning mechanisms, we presented results comparing pre-pruning to post-pruning when both are used in conjunction with the parametric chi-squared test. The results show that the two strategies perform comparably on the practical datasets used in this thesis and differ mainly in the significance level that is required for optimum performance. Thus the potentially negative effect of attribute interactions on the performance of pre-pruning does not seem to be problematic in many practical learning problems.

The last set of experiments examined pruning procedures for decision lists. We discussed drawbacks of global optimization approaches to this problem and illustrated how standard incremental pruning, although it avoids the drawbacks of global pruning, can sometimes prune too aggressively. We proposed tree-based incremental pruning as a way of avoiding the problems associated with global optimization and the overpruning phenomenon in the basic incremental approach. It was shown empirically that the tree-based method produces more accurate rule sets than the standard incremental algorithm. The resulting rule sets are larger but their size can be reduced effectively with significance-based rule selection and pruning. Finally, we demonstrated how the induction process can be accelerated by building partial decision trees instead of full ones. This decreases the runtime of the rule inducer if the domain requires little pruning. However, it also increases the rule sets' size. The resulting algorithm compares favorably to the state-of-the-art rule learners RIPPER and C4.5.

6.2 Contributions

This thesis makes several contributions to research. Listed roughly in order of importance, these contributions are:

- the finding that the significance level for pruning must be adjusted to fit the domain (Chapters 3, 4, and 5) and the significance test employed (Chapters 3 and 4);
- empirical evidence that, despite their theoretical advantages, permutation tests do not outperform their parametric counterparts in practical pruning problems (Chapters 3 and 4);
- an algorithm for tree-based incremental pruning that eliminates overpruning problems in incremental pruning methods for rule learners (Chapter 5);

- a mechanism for avoiding overfitting in reduced-error pruning by incorporating significance tests in the pruning process (Chapter 3);
- empirical evidence that cross-validation can be used to find an appropriate significance level for pruning automatically (Chapter 3);
- empirical evidence that, in practice, the horizon effect can generally not be held responsible for potential performance differences between pre- and post-pruning on practical datasets (Chapter 4);
- the finding that standard incremental pruning mechanisms for rule learners can lead to hasty generalization (Chapter 5);
- an algorithm for inducing partial trees that reduces the computational complexity of tree-based incremental pruning (Chapter 5);
- proof that a rule can be induced in time that is linear in the number of training instances (Chapter 5);
- a simple quantitative relationship explaining when the classification error of a disjunct increases with its size if no search is involved in generating it (Chapter 2);
- a simple experiment demonstrating the effect of search on the error rate of a disjunct (Chapter 2);
- empirical evidence that significance-based rule selection produces more comprehensible decision lists than error-based selection (Chapter 5);
- empirical evidence that it is not necessary to locally adjust for significance tests on multiple attributes when using pre-pruning on practical datasets (Chapter 4);
- empirical evidence that incremental pruning algorithms for rule learners can compete with state-of-the-art approaches that use global optimization (Chapter 5).

6.3 Future work

Several topics for future research are raised by the issues addressed in this thesis. An interesting question is whether the methods from Chapter 3 can be used to successfully implement "edge-based pruning" in decision trees (Pereira & Singer, 1999). Standard pruning algorithms for decision trees do not consider whether it is best to eliminate only some of the branches that extend from a node, they only test whether deleting all of them is likely to improve performance. By removing unreliable branches individually, it is possible to fine-tune the pruning process and avoid modifications that are unnecessarily drastic. The resulting decision tree differs from a "normal" decision tree in that an internal node is used to classify a test instance if it cannot be filtered further down the tree. However, this is only a minor modification in the way the tree is interpreted and does not make the classification process harder to understand.

Another potential application for the methods in Chapter 3 is the problem of global attribute selection where only a subset of the original attributes is given to the learning scheme. This is a way of "pruning" the instance space so that the learning algorithm is less likely to overfit the training data. The "wrapper" method for attribute selection attempts to find the subset of attributes that produces the classifier with the greatest estimated accuracy (Kohavi, 1995b). For each subset of attributes that is evaluated, it performs a cross-validation of the learning scheme to estimate the resulting classifier's performance. This is done in conjunction with a best-first search through the space of possible attribute subsets (Kohavi, 1995b). The runtime of this procedure is at least quadratic in the number of attributes and is therefore usually very slow. However, if the attributes can be ranked effectively according to how predictive they are, the process can be accelerated by starting with the most predictive attribute and adding one attribute at a time until all attributes are used up, each time evaluating the resulting classifier by cross-validation. The runtime of this procedure is linear in the number of attributes. A promising approach is to rank the attributes according to their significance and to let cross-validation automatically select the appropriate significance level for deciding whether an attribute is included in the final subset—like the procedure for selecting a significance level in

Chapter 3.

An issue raised by the results of Chapter 5 is whether the standard reduced-error pruning technique can be modified so that, even in domains where only little pruning is required, it is at least as accurate as methods that do not have to split off a pruning set—for example, C4.5's error-based pruning. It may be possible to achieve this by varying the percentage of the full training set that is set aside for pruning, or by abstaining from pruning if both the pruned and the unpruned model have the same error rate on the pruning data.

As mentioned in Chapter 5, tree-based pruning can be used to generate pruned rules for boosting (Freund & Schapire, 1996). Overpruning is not necessarily a problem in this situation because the reweighting mechanism employed by boosting algorithms automatically places more weight on misclassified instances. Therefore boosting can recover from overpruning if additional boosting iterations are performed. However, tree-based pruning can potentially reduce the number of iterations that are required to obtain a certain level of accuracy. Moreover, as in the induction of decision lists, it can prevent boosting from stopping too early due to a rule that is not sufficiently accurate.

This thesis has investigated the usefulness of significance tests in pruning algorithms for decision trees and lists. It is clear that there are many other applications of these statistical tools in learning algorithms, some of which have been discussed in the sections on related work at the end of each chapter. Several of the findings presented in this thesis—for example, the importance of an appropriately chosen significance level—are directly applicable to these problems. They are likely to prove useful in many practical data mining applications—whether they are based on decision trees or lists, or some other technique that is particularly suitable for the specific problem at hand.

Appendix A

Results for post-pruning

A.1 Parametric vs. non-parametric tests

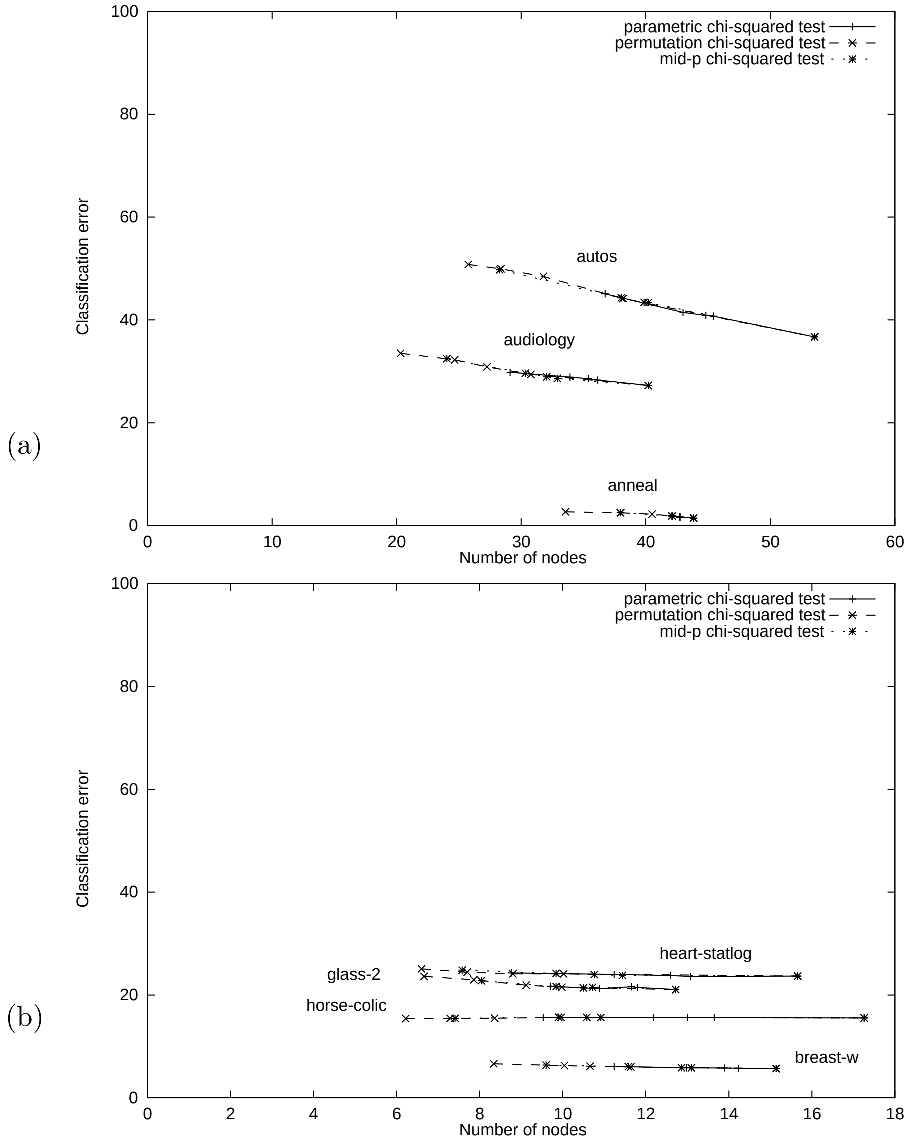
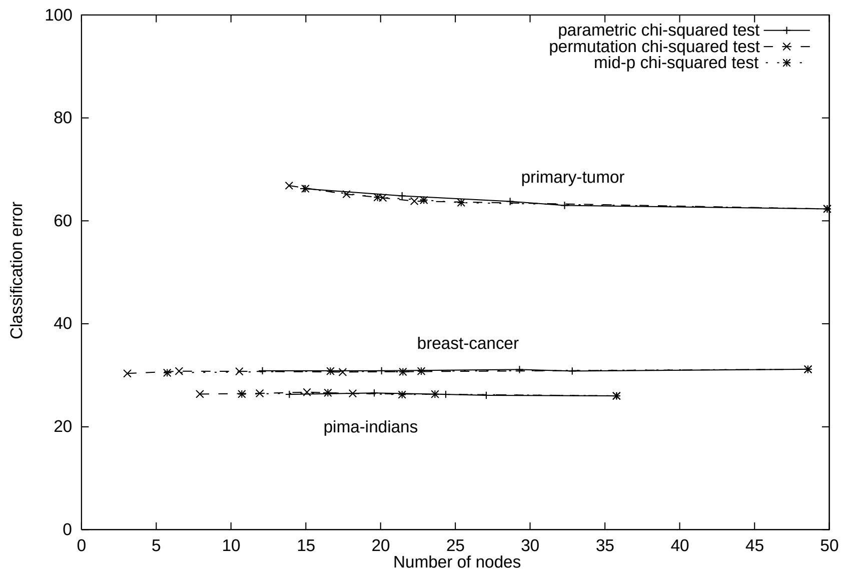
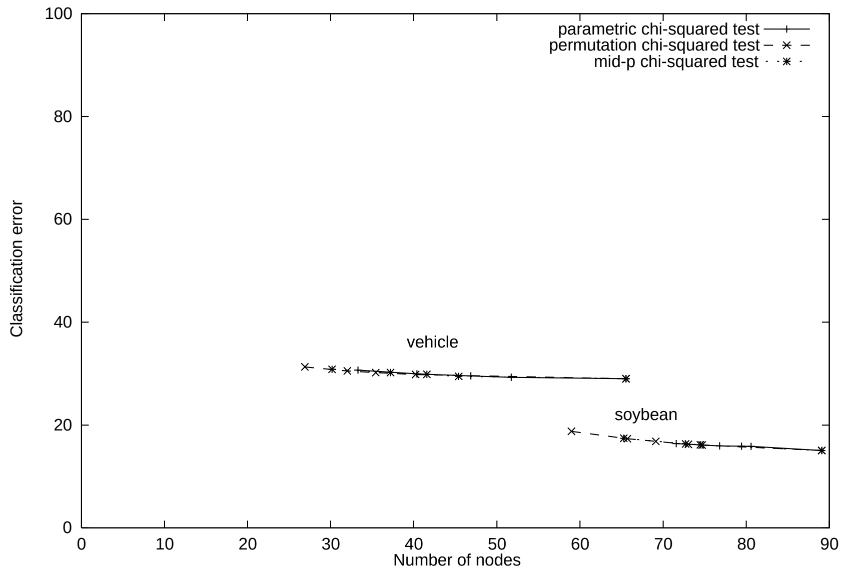


Figure A.1: Comparison of significance tests

(c)



(d)



(e)

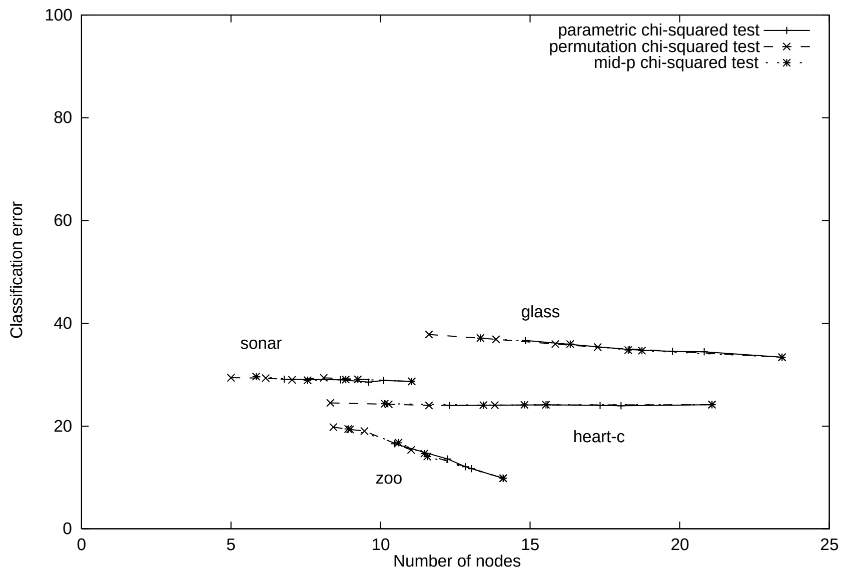
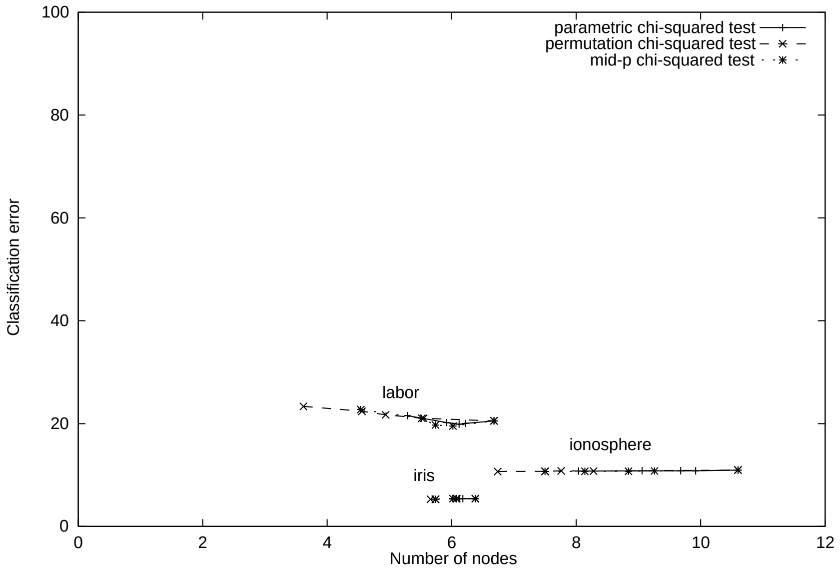
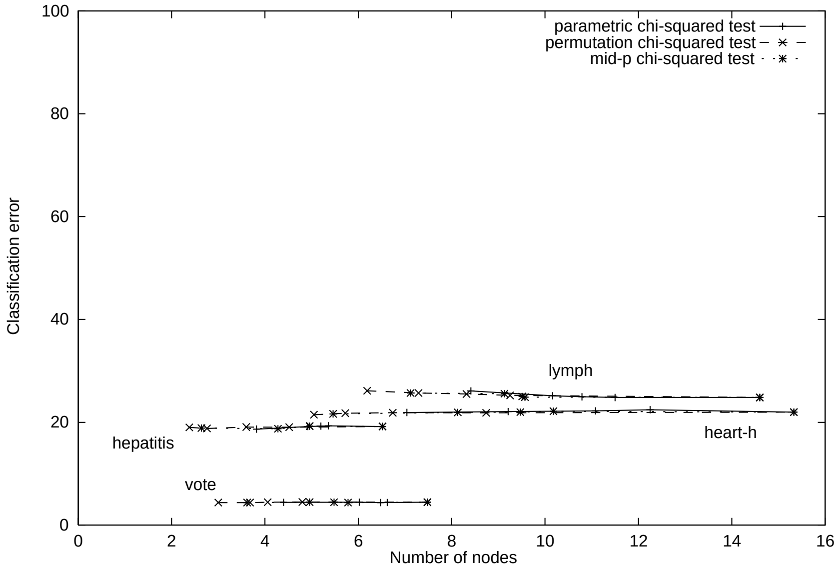


Figure A.1: Comparison of significance tests (continued)

(f)



(g)



(h)

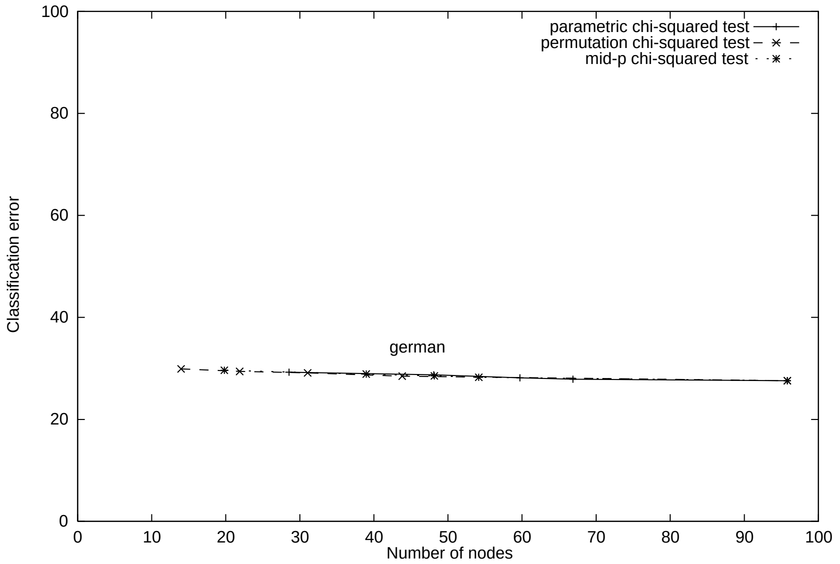


Figure A.1: Comparison of significance tests (continued)

A.2 Reduced-overlap pruning

Table A.1: Accuracy and tree sizes for reduced-error pruning (REP) compared to reduced-overlap pruning (ROP)

	Accuracy			Tree sizes		
Dataset	REP	ROP		REP	ROP	
anneal	98.5±0.4	97.0±0.6	•	43.8±2.7	36.1±1.8	○
audiology	72.8±1.5	69.1±1.1	•	40.2±2.6	28.7±2.5	○
australian	84.3±0.9	84.9±0.7		34.3±7.9	14.5±6.8	○
autos	63.3±2.8	60.4±3.8	•	53.6±2.8	47.1±2.3	○
balance-scale	78.4±0.9	75.5±1.1	•	54.6±3.2	32.3±2.3	○
breast-cancer	68.8±1.6	69.0±1.0		48.6±5.3	23.2±7.9	○
breast-w	94.3±0.7	93.7±0.5	•	15.1±1.5	9.9±1.2	○
german	72.4±1.4	71.8±1.3	•	95.8±9.5	57.5±9.1	○
glass-2	79.0±2.3	77.0±2.1		12.7±2.3	7.8±1.8	○
glass	66.6±3.4	63.1±3.7	•	23.4±1.8	16.0±2.0	○
heart-c	75.8±2.3	76.1±2.8		21.1±1.6	13.5±1.7	○
heart-h	78.0±1.7	77.3±1.4		15.3±1.8	8.4±1.5	○
heart-statlog	76.3±2.8	75.8±2.4		15.7±1.6	9.5±1.5	○
hepatitis	80.8±2.5	80.3±2.5		6.5±1.9	3.2±1.0	○
horse-colic	84.5±0.8	83.8±1.0		17.3±3.9	9.8±2.5	○
ionosphere	89.0±0.8	88.4±1.7		10.6±2.1	6.9±0.6	○
iris	94.6±0.8	94.8±0.8		6.4±0.5	5.8±0.3	○
labor	79.5±3.9	78.1±5.6		6.7±0.8	4.8±1.3	○
lymphography	75.2±1.7	74.4±1.7		14.6±3.0	10.7±2.1	○
pima-indians	74.0±0.6	74.1±0.7		35.8±5.4	13.1±2.0	○
primary-tumor	37.7±1.9	35.3±1.7	•	49.9±4.7	23.6±3.6	○
sonar	71.3±2.4	70.3±2.5		11.0±1.3	6.4±1.0	○
soybean	85.0±0.9	82.7±1.3	•	89.1±1.9	72.3±2.7	○
vehicle	71.0±1.1	69.3±0.9	•	65.5±4.5	35.4±1.9	○
vote	95.6±0.5	95.6±0.1		7.5±1.2	4.0±0.7	○
vowel	71.6±1.5	63.5±1.3	•	280.9±4.3	249.4±4.0	○
zoo	90.2±2.3	87.0±3.2	•	14.1±0.4	12.3±0.7	○

Table A.2: Results of paired *t*-tests (p=0.05): number indicates how often method in column significantly outperforms method in row

	Accuracy		Tree sizes	
	REP	ROP	REP	ROP
REP	–	0	–	27
ROP	12	–	0	–

Appendix B

Results for pre-pruning

B.1 Parametric vs. non-parametric tests

(a)

(b)

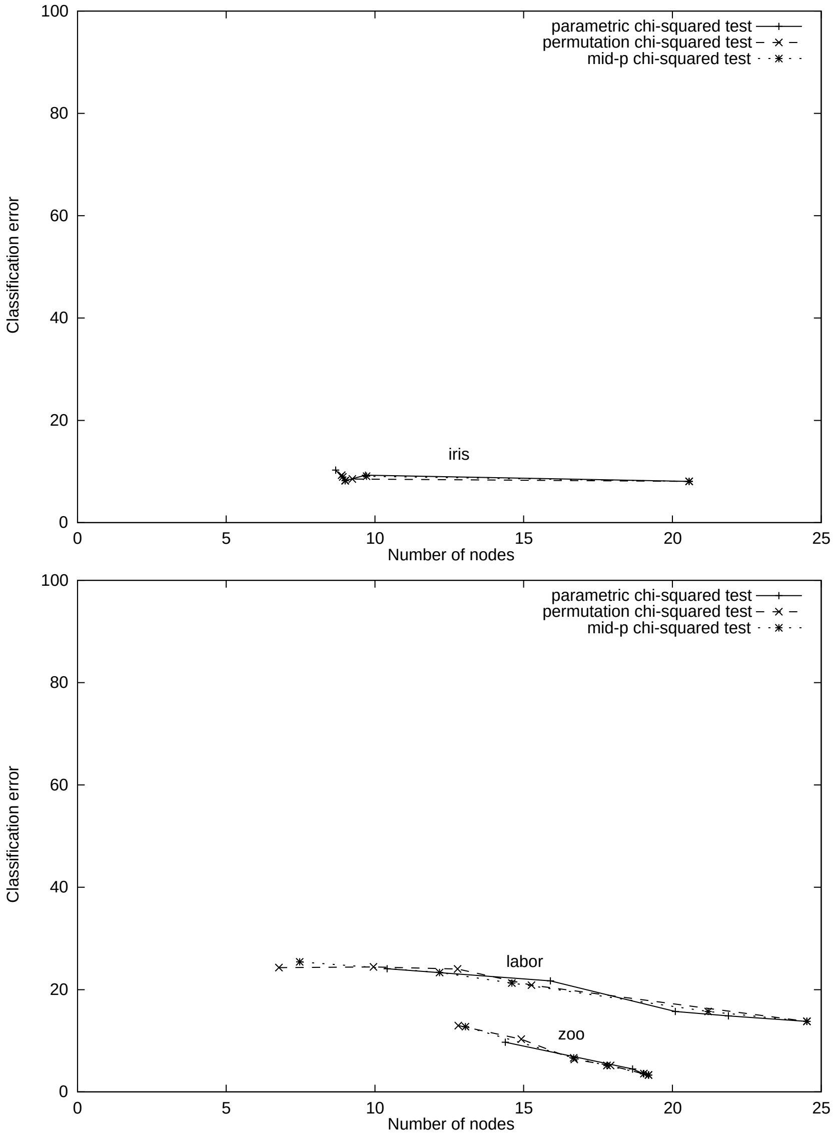
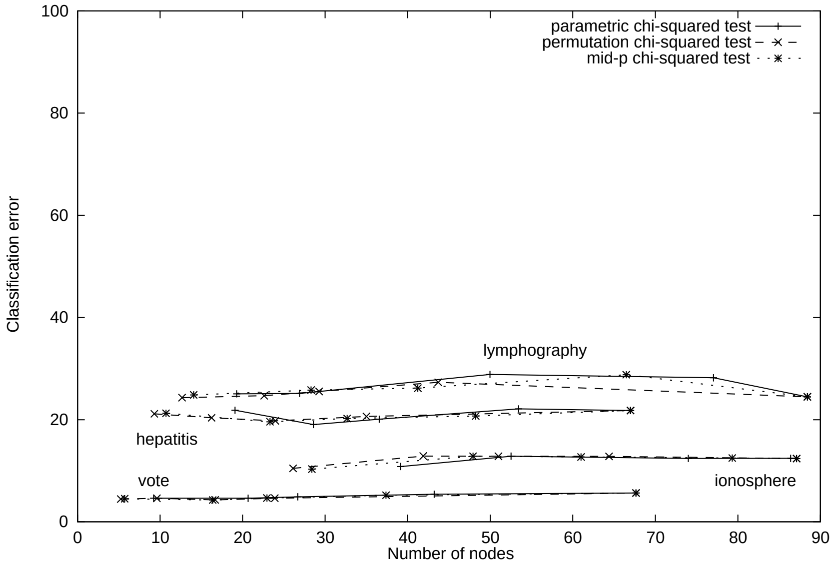
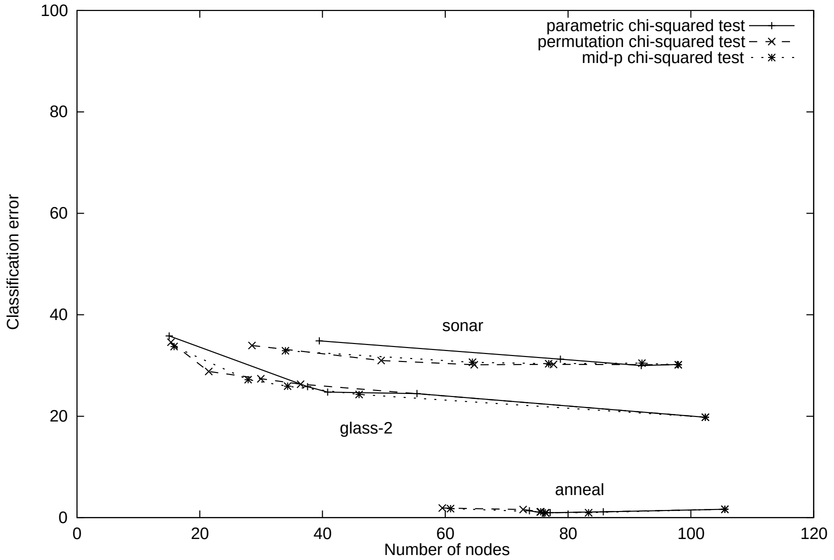


Figure B.1: Comparison of significance tests

(c)



(d)



(e)

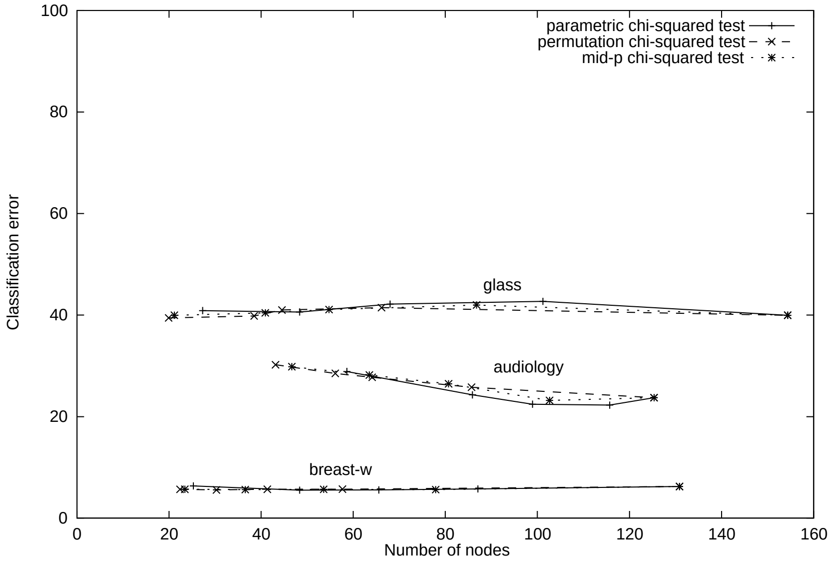
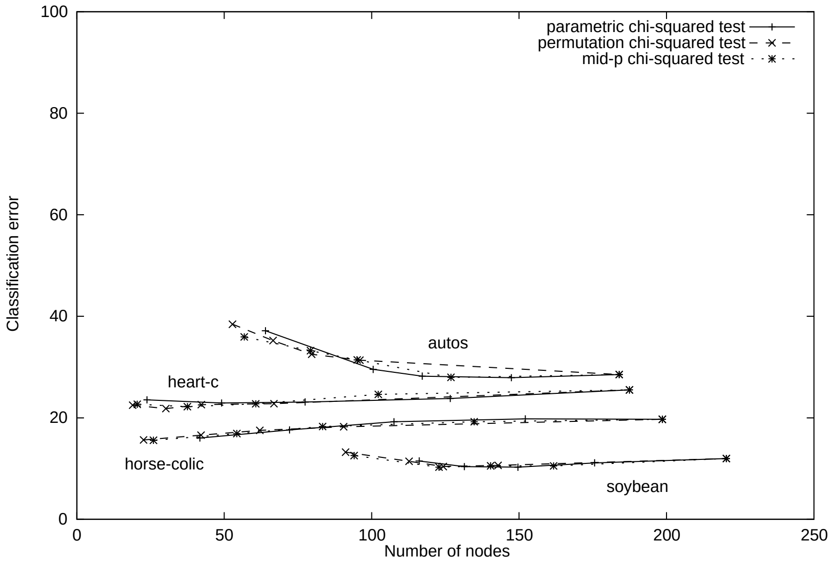
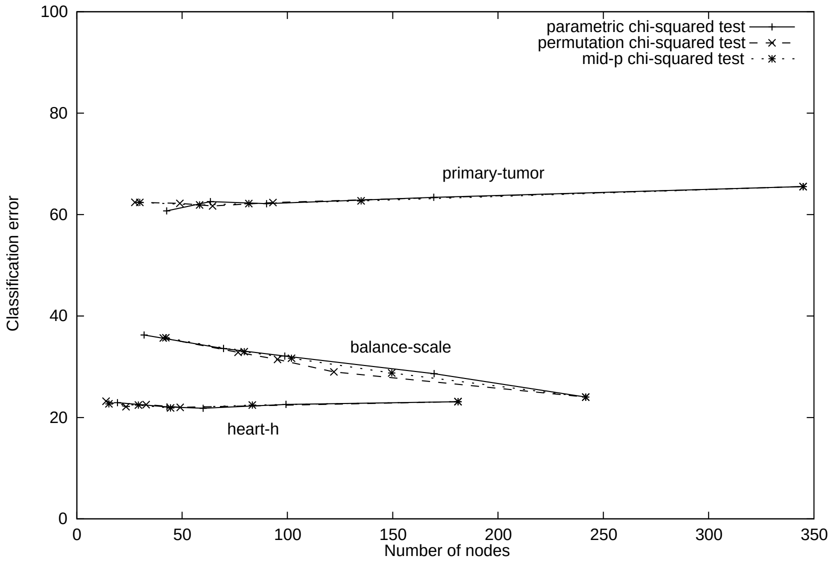


Figure B.1: Comparison of significance tests (continued)

(f)



(g)



(h)

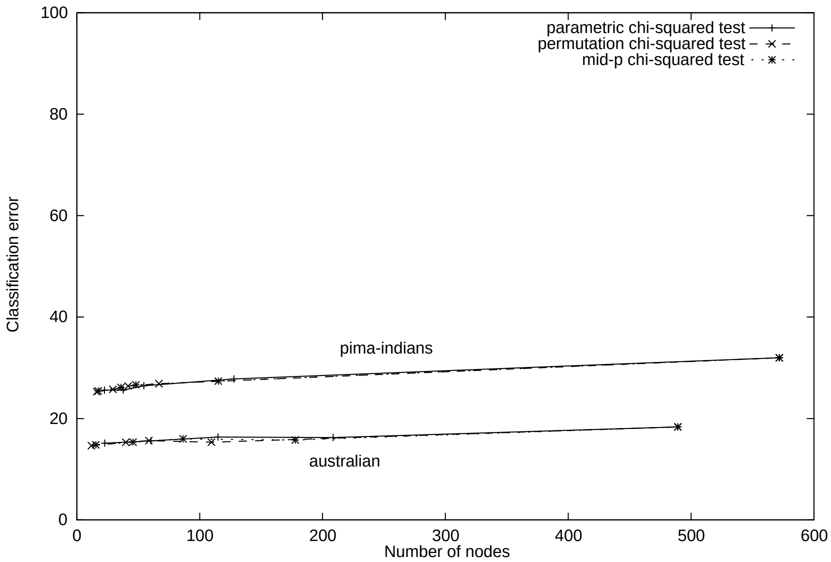


Figure B.1: Comparison of significance tests (continued)

(i)

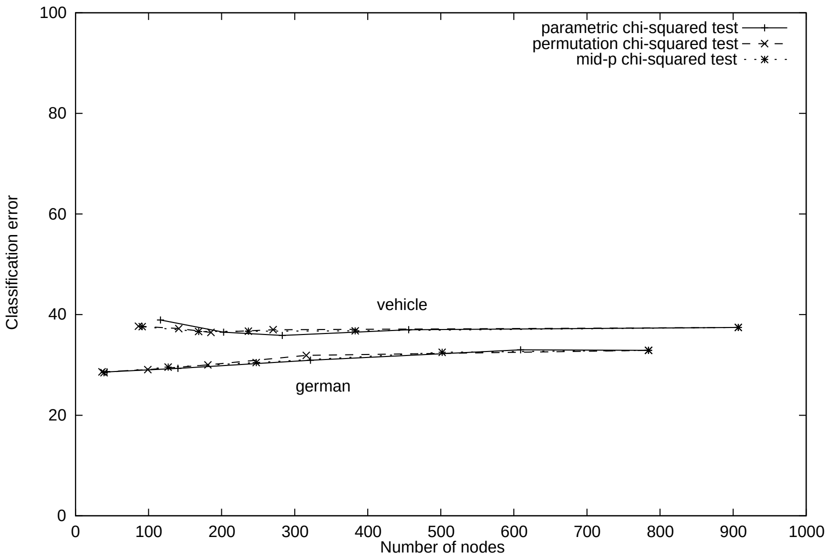


Figure B.1: Comparison of significance tests (continued)

B.2 Adjusting for multiple tests

(a)

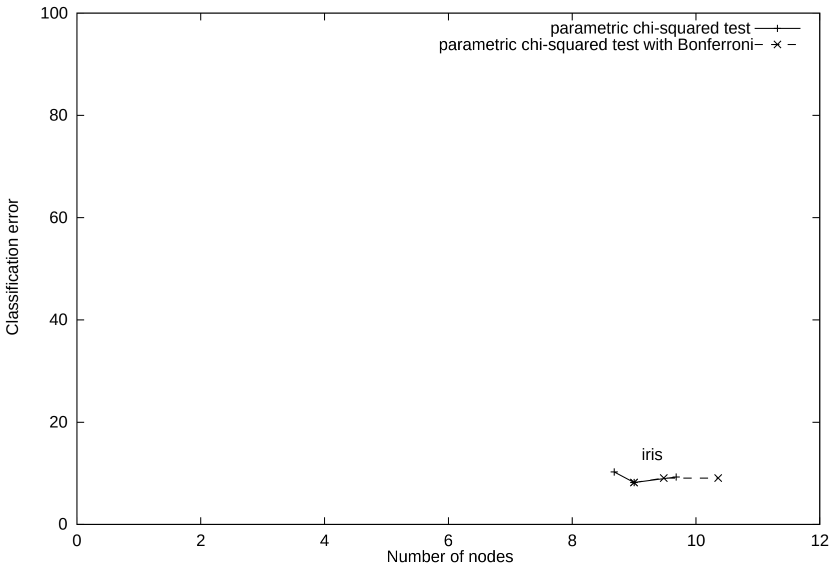
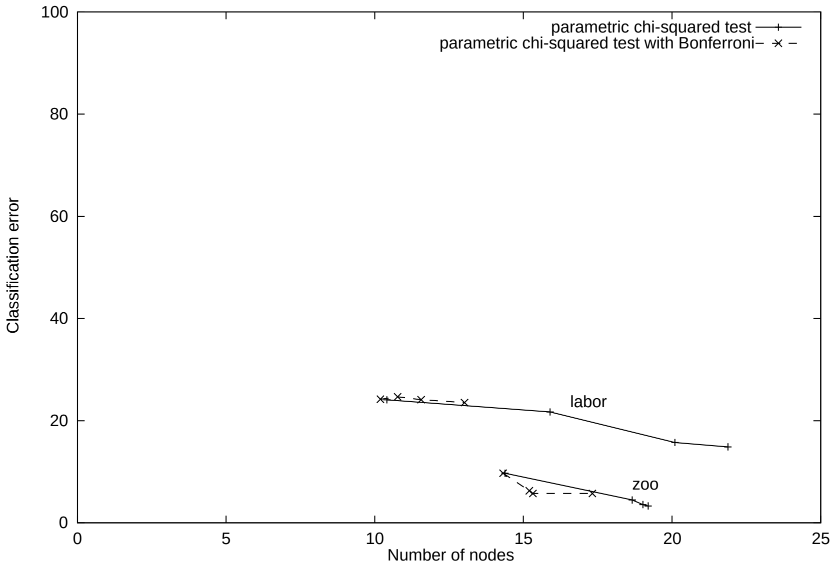
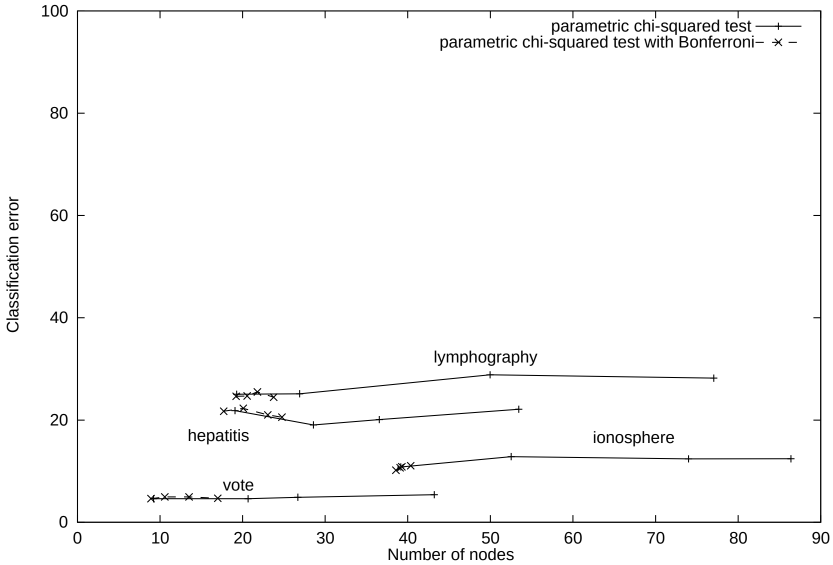


Figure B.2: Using the Bonferroni correction

(b)



(c)



(d)

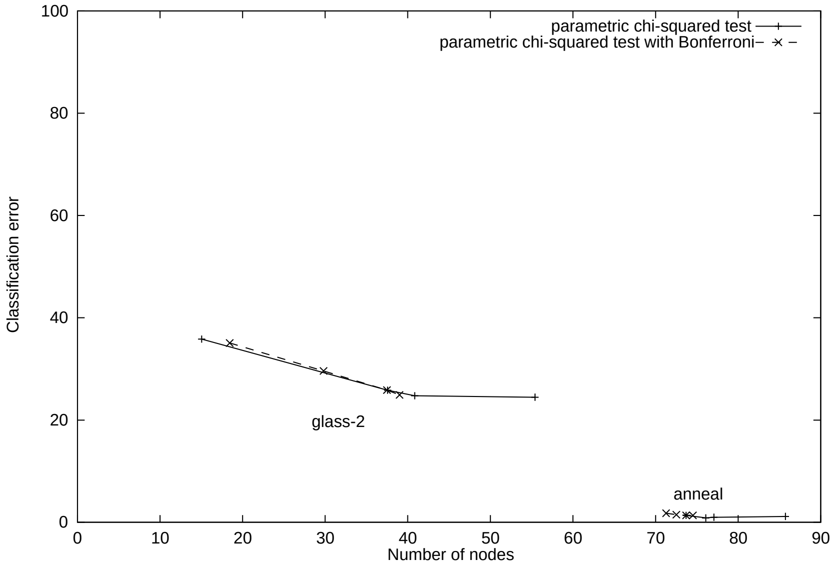
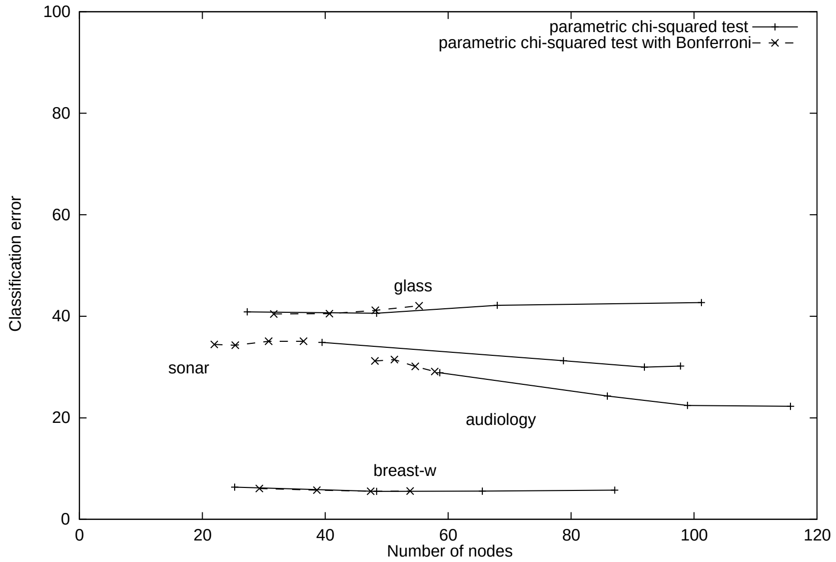
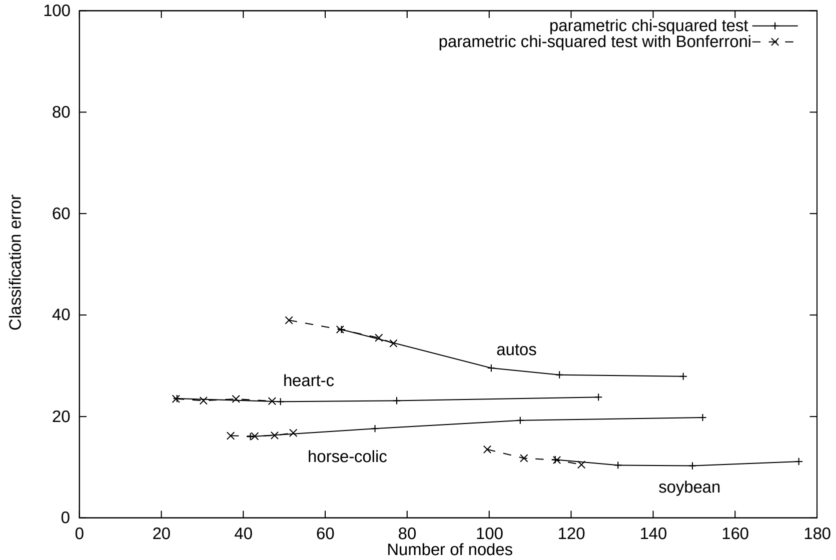


Figure B.2: Using the Bonferroni correction (continued)

(e)



(f)



(g)

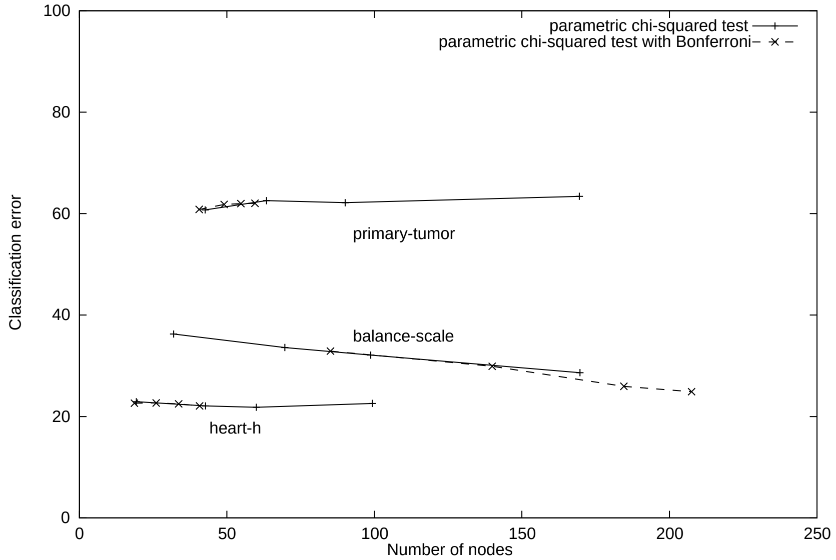
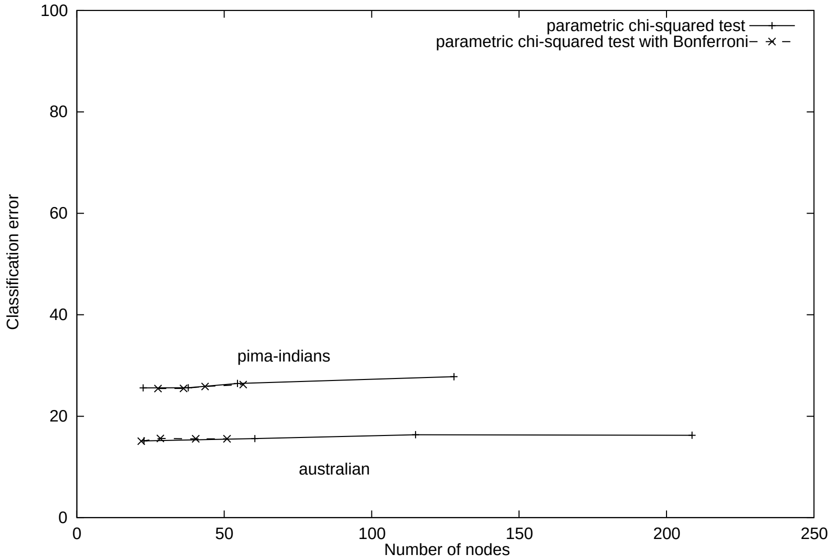


Figure B.2: Using the Bonferroni correction (continued)

(h)



(i)

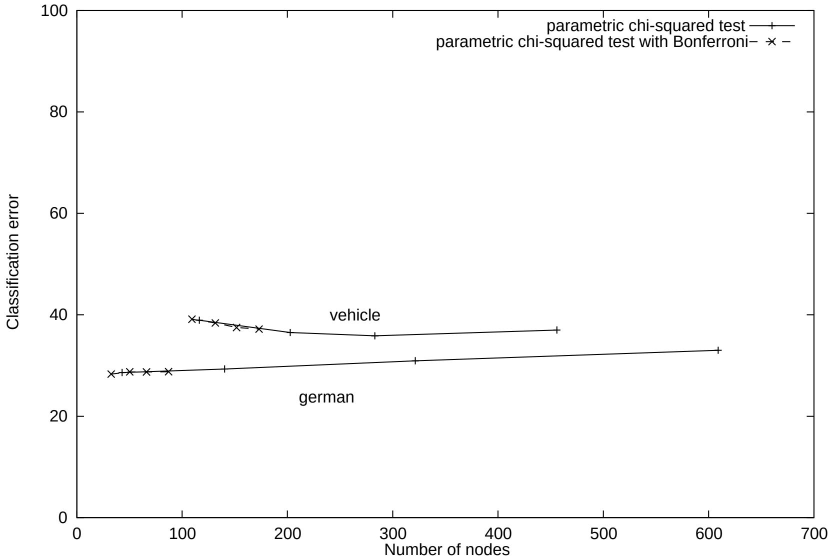


Figure B.2: Using the Bonferroni correction (continued)

B.3 Pre- vs. post-pruning

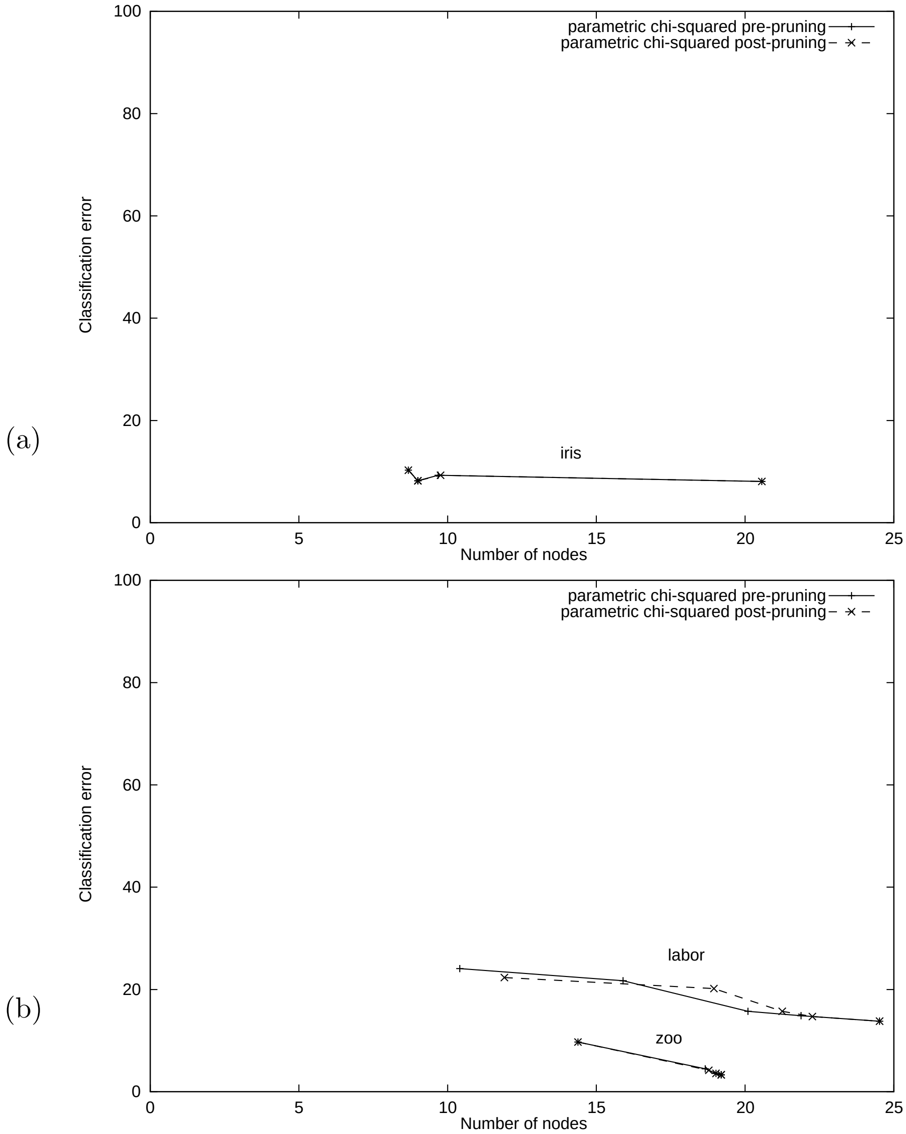
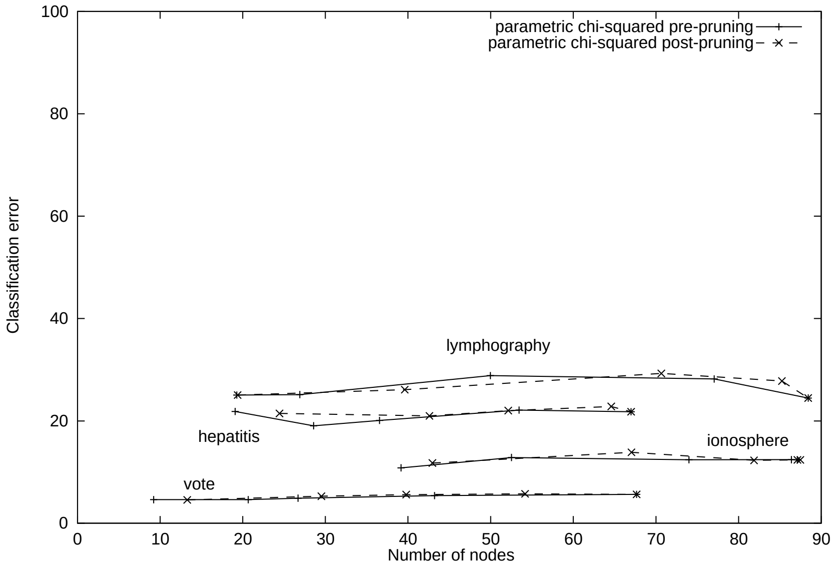
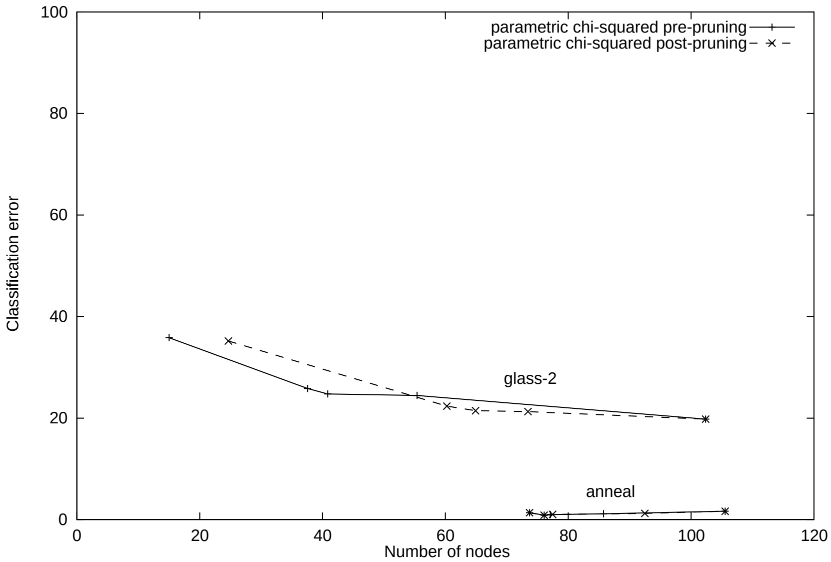


Figure B.3: Pre- vs. post-pruning

(c)



(d)



(e)

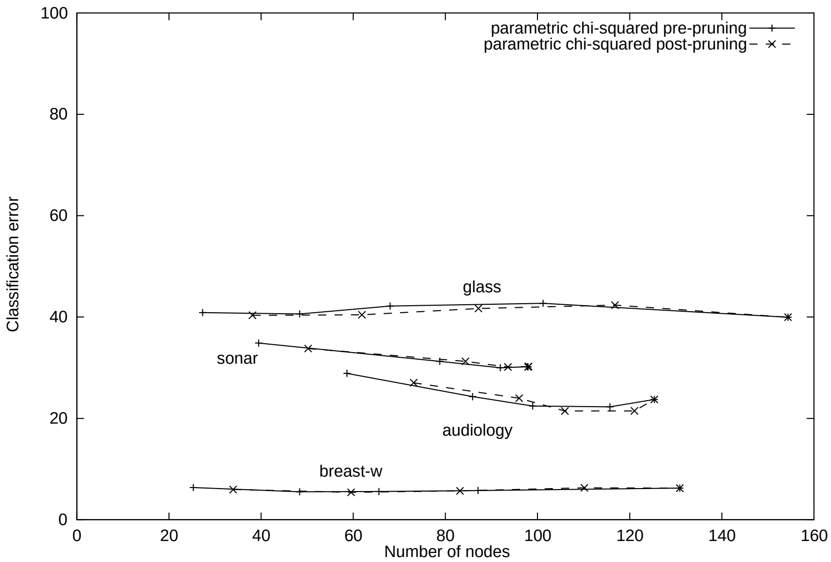
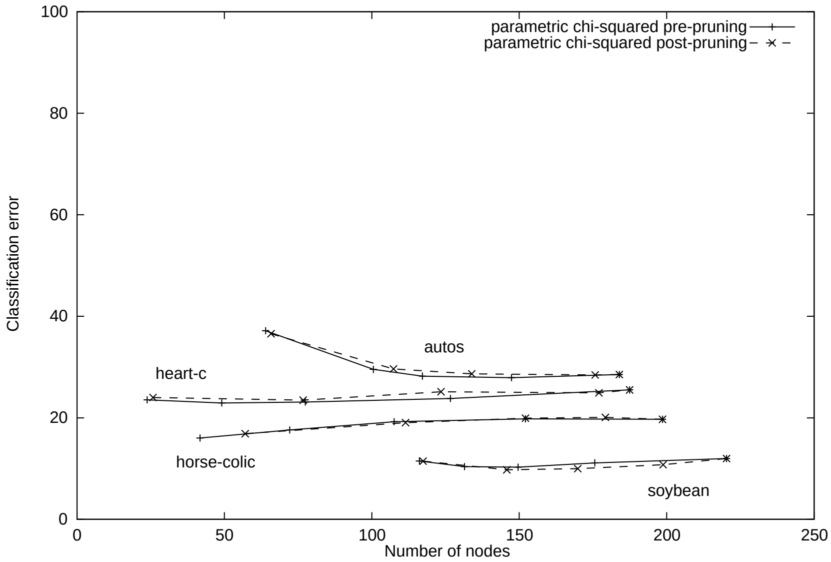
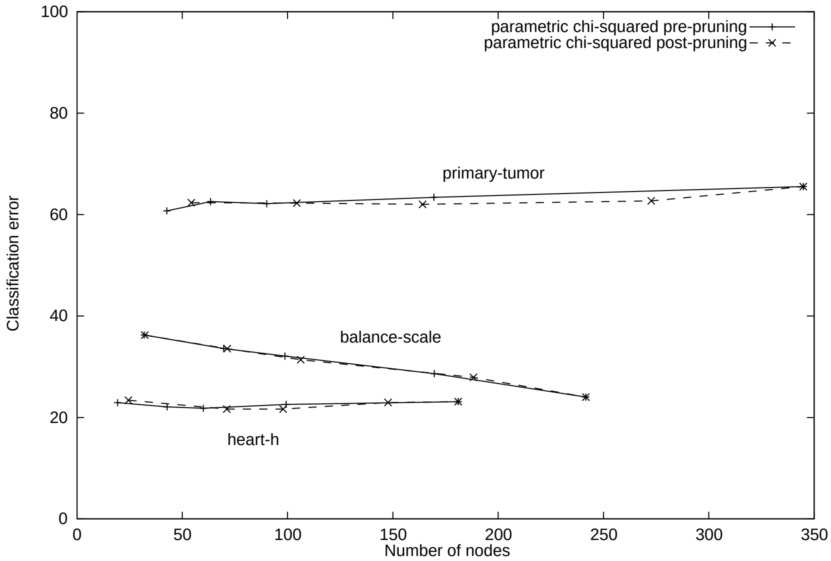


Figure B.3: Pre- vs. post-pruning (continued)

(f)



(g)



(h)

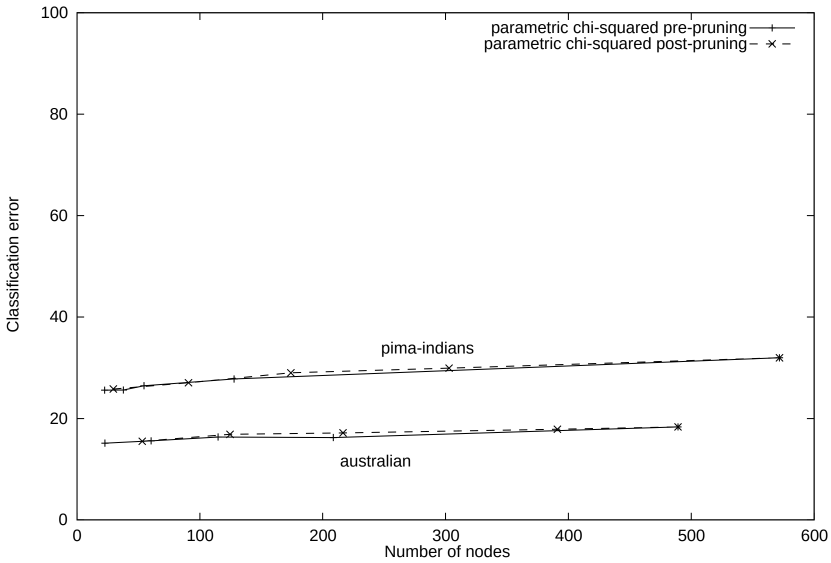


Figure B.3: Pre- vs. post-pruning (continued)

(i)

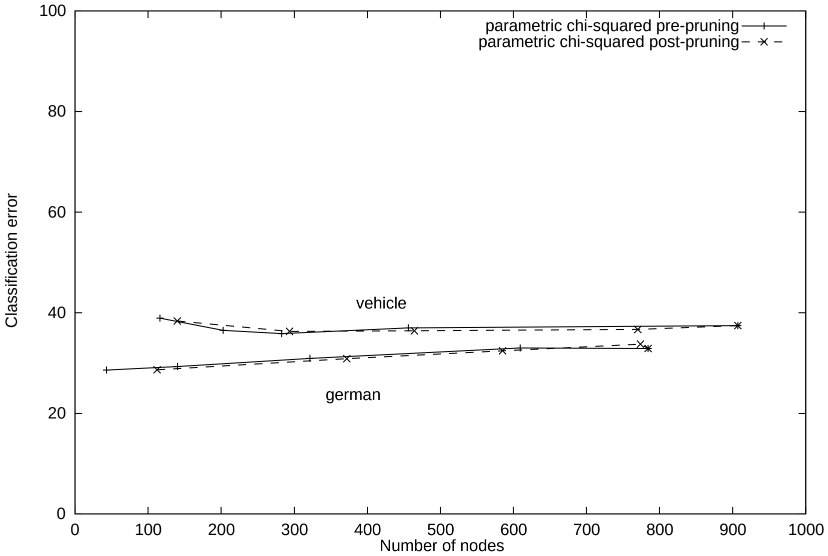


Figure B.3: Pre- vs. post-pruning (continued)

Appendix C

Results for pruning decision lists

Table C.1: Accuracy and number of rules for PART_{sig}^{var} compared to C4.5

	Accuracy			Number of rules		
Dataset	PART_{sig}^{var}	C4.5		PART_{sig}^{var}	C4.5	
anneal	98.3±0.3	98.6±0.2	∘	14.0±0.4	17.2±0.4	•
audiology	75.0±1.3	76.3±1.0	∘	17.8±0.6	21.4±0.3	•
australian	84.3±1.4	84.9±1.1		3.9±0.3	14.4±0.8	•
autos	70.6±2.1	76.7±2.6	∘	20.9±1.0	24.8±0.8	•
balance-scale	81.5±1.2	77.9±0.6	•	39.2±1.0	37.1±0.8	∘
breast-cancer	72.0±1.2	70.4±1.6	•	10.2±1.4	10.0±0.6	
breast-w	93.8±0.9	95.5±0.6	∘	4.1±0.3	10.1±0.4	•
german	70.9±0.7	71.5±0.7		6.6±0.8	27.2±1.5	•
glass-2	78.2±3.1	79.6±2.2		3.9±0.2	9.6±0.6	•
glass	66.6±1.4	66.6±3.0		13.1±0.4	14.6±0.6	•
heart-c	78.1±1.3	80.2±1.4	∘	4.5±0.5	14.2±0.7	•
heart-h	79.4±1.4	79.5±1.4		7.4±0.7	8.3±0.5	•
heart-statlog	78.9±2.4	81.1±1.4	∘	4.1±0.3	12.0±0.8	•
hepatitis	81.9±2.2	79.6±0.9	•	8.2±0.4	9.2±0.4	•
horse-colic	85.2±0.5	83.0±0.6	•	3.6±0.3	7.0±0.3	•
ionosphere	89.0±1.0	90.5±1.7	∘	3.0±0.2	10.6±0.5	•
iris	95.1±0.6	95.0±1.0		3.8±0.2	5.0±0.1	•
labor	83.3±5.3	81.4±2.6		3.0±0.3	5.2±0.3	•
lymphography	76.7±3.5	77.7±1.9		11.2±0.6	11.3±0.3	
pima-indians	73.5±1.1	74.2±1.1		4.7±0.2	11.3±0.8	•
primary-tumor	40.4±1.6	40.6±1.7		41.8±0.9	18.3±0.7	∘
sonar	70.1±2.4	74.5±3.2	∘	4.3±0.2	9.6±0.3	•
soybean	91.1±0.9	91.8±0.7		39.2±0.9	35.9±0.5	∘
vehicle	69.2±1.8	72.8±1.0	∘	12.7±0.7	29.0±2.3	•
vote	95.7±0.2	95.8±0.6		3.0±0.2	7.0±0.2	•
vowel	72.8±0.9	78.1±1.2	∘	56.5±1.5	73.9±1.8	•
zoo	90.4±1.3	91.6±1.2	∘	7.3±0.3	9.1±0.1	•

Table C.2: Results of paired t-tests (p=0.05): number indicates how often method in column significantly outperforms method in row

	Accuracy		Number of rules	
	PART_{sig}^{var}	C4.5	PART_{sig}^{var}	C4.5
PART_{sig}^{var}	–	11	–	3
C4.5	4	–	22	–

Table C.3: Accuracy and number of rules for RIPPER compared to C4.5

	Accuracy			Number of rules		
Dataset	RIPPER	C4.5		RIPPER	C4.5	
anneal	98.3±0.1	98.6±0.2	○	8.7±0.2	17.2±0.4	•
audiology	72.3±2.2	76.3±1.0	○	17.0±0.5	21.4±0.3	•
australian	85.3±0.7	84.9±1.1		4.1±0.5	14.4±0.8	•
autos	72.0±2.0	76.7±2.6	○	12.4±0.5	24.8±0.8	•
balance-scale	81.0±1.1	77.9±0.6	•	11.3±0.5	37.1±0.8	•
breast-cancer	71.8±1.6	70.4±1.6	•	3.2±0.6	10.0±0.6	•
breast-w	95.6±0.6	95.5±0.6		5.6±0.3	10.1±0.4	•
german	71.4±0.7	71.5±0.7		4.1±0.5	27.2±1.5	•
glass-2	80.9±1.4	79.6±2.2		4.4±0.4	9.6±0.6	•
glass	66.7±2.1	66.6±3.0		8.1±0.4	14.6±0.6	•
heart-c	78.5±1.9	80.2±1.4		4.3±0.2	14.2±0.7	•
heart-h	78.7±1.3	79.5±1.4		3.3±0.2	8.3±0.5	•
heart-statlog	79.0±1.4	81.1±1.4	○	4.0±0.2	12.0±0.8	•
hepatitis	77.2±2.0	79.6±0.9	○	2.9±0.1	9.2±0.4	•
horse-colic	85.0±0.8	83.0±0.6	•	3.8±0.4	7.0±0.3	•
ionosphere	89.2±0.8	90.5±1.7	○	5.0±0.5	10.6±0.5	•
iris	94.4±1.7	95.0±1.0		3.5±0.1	5.0±0.1	•
labor	83.5±3.9	81.4±2.6		3.3±0.1	5.2±0.3	•
lymphography	76.1±2.4	77.7±1.9		6.2±0.2	11.3±0.3	•
pima-indians	75.2±1.1	74.2±1.1		3.8±0.4	11.3±0.8	•
primary-tumor	38.5±0.8	40.6±1.7	○	8.9±0.6	18.3±0.7	•
sonar	75.7±1.9	74.5±3.2		4.7±0.3	9.6±0.3	•
soybean	92.1±0.5	91.8±0.7		27.4±0.8	35.9±0.5	•
vehicle	69.0±0.6	72.8±1.0	○	14.5±0.5	29.0±2.3	•
vote	95.6±0.3	95.8±0.6		2.6±0.2	7.0±0.2	•
vowel	69.6±1.9	78.1±1.2	○	41.5±1.0	73.9±1.8	•
zoo	87.8±2.4	91.6±1.2	○	7.0±0.2	9.1±0.1	•

Table C.4: Results of paired t-tests (p=0.05): number indicates how often method in column significantly outperforms method in row

	Accuracy		Number of rules	
	RIPPER	C4.5	RIPPER	C4.5
RIPPER	–	10	–	0
C4.5	3	–	27	–

Table C.5: Accuracy and number of rules for PART_{C4} compared to RIPPER.

	Accuracy			Number of rules		
Dataset	PART_{C4}	RIPPER		PART_{C4}	RIPPER	
anneal	98.4±0.3	98.3±0.1		14.7±0.3	8.7±0.2	○
audiology	78.7±1.0	72.3±2.2	•	19.6±0.3	17.0±0.5	○
australian	84.3±1.2	85.3±0.7		30.7±1.5	4.1±0.5	○
autos	74.5±1.1	72.0±2.0	•	20.5±0.8	12.4±0.5	○
balance-scale	82.3±1.2	81.0±1.1	•	38.6±1.0	11.3±0.5	○
breast-cancer	69.6±1.6	71.8±1.6	○	19.1±1.2	3.2±0.6	○
breast-w	94.9±0.4	95.6±0.6	○	10.0±0.5	5.6±0.3	○
german	70.0±1.4	71.4±0.7	○	69.6±1.3	4.1±0.5	○
glass-2	80.0±4.0	80.9±1.4		6.7±0.4	4.4±0.4	○
glass	69.8±2.3	66.7±2.1	•	15.3±0.6	8.1±0.4	○
heart-c	78.5±1.7	78.5±1.9		19.7±0.6	4.3±0.2	○
heart-h	80.5±1.5	78.7±1.3	•	8.8±0.6	3.3±0.2	○
heart-statlog	78.9±1.3	79.0±1.4		17.8±1.0	4.0±0.2	○
hepatitis	80.2±1.9	77.2±2.0	•	8.4±0.5	2.9±0.1	○
horse-colic	84.4±0.8	85.0±0.8		9.5±0.8	3.8±0.4	○
ionosphere	90.6±1.3	89.2±0.8	•	7.5±0.5	5.0±0.5	○
iris	93.7±1.6	94.4±1.7		4.0±0.4	3.5±0.1	○
labor	77.3±3.9	83.5±3.9	○	3.6±0.2	3.3±0.1	○
lymphography	76.5±2.7	76.1±2.4		11.6±0.5	6.2±0.2	○
pima-indians	73.6±0.5	75.2±1.1	○	7.4±0.4	3.8±0.4	○
primary-tumor	41.7±1.3	38.5±0.8	•	40.8±1.0	8.9±0.6	○
sonar	76.5±2.3	75.7±1.9		7.5±0.4	4.7±0.3	○
soybean	91.4±0.5	92.1±0.5	○	37.0±0.8	27.4±0.8	○
vehicle	72.4±0.8	69.0±0.6	•	33.3±1.2	14.5±0.5	○
vote	95.9±0.6	95.6±0.3		6.2±0.4	2.6±0.2	○
vowel	78.1±1.1	69.6±1.9	•	65.9±1.5	41.5±1.0	○
zoo	92.2±1.2	87.8±2.4	•	7.6±0.1	7.0±0.2	○

Table C.6: Results of paired t-tests (p=0.05): number indicates how often method in column significantly outperforms method in row

	Accuracy		Number of rules	
	PART_{C4}	RIPPER	PART_{C4}	RIPPER
PART_{C4}	–	6	–	27
RIPPER	11	–	0	–

Bibliography

Agresti, A. (1990). *Categorical Data Analysis*. New York: John Wiley & Sons.

Aha, D., Kibler, D. & Albert, M. (1991). Instance-based learning algorithms. *Machine Learning*, *6(1)*, 37–66.

Ali, K. M. & Pazzani, M. J. (1995). *Reducing the small disjuncts problem by learning probabilistic concept descriptions*, Volume III: Selecting Good Models, chapter 10, (pp. 183–199). MIT Press, Cambridge, MA.

Almuallim, H. (1996). An efficient algorithm for optimal pruning of decision trees. *Artificial Intelligence*, *83(2)*, 347–362.

Blake, C., Keogh, E. & Merz, C. (1998). UCI repository of machine learning databases. Technical report, University of California, Department of Information and Computer Science, Irvine, CA. [http://www.ics.uci.edu/~mlearn/MLRepository.html].

Bohanec, M. & Bratko, I. (1994). Trading accuracy for simplicity in decision trees. *Machine Learning*, *15(3)*, 223–250.

Bradford, J. P., Kunz, C., Kohavi, R., Brunk, C. & Brodley, C. E. (1998). Pruning decision trees with misclassification costs. In Nédellec, C. & Rouveirol, C. (Eds.), *Proc of the 10th European Conf on Machine Learning* (pp. 131–136). Chemnitz, Germany: Springer-Verlag, Berlin.

Breiman, L. (1996). Bias, variance, and arcing classifiers. Technical report, Dept. of Statistics, University of California.

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

Breslow, L. A. & Aha, D. W. (1997a). Comparing tree-simplification procedures. In *Proc of the 6th Int Workshop on AI & Statistics*. Ft. Lauderdale, Florida: unpublished.

Breslow, L. A. & Aha, D. W. (1997b). Simplifying decision trees: A survey. *Knowledge Engineering Review*, *12(1)*, 1–40.

Brodley, C. E. & Utgoff, P. E. (1995). Multivariate decision trees. *Machine Learning*, *19(1)*, 45–77.

Brunk, C. A. & Pazzani, M. J. (1991). An investigation of noise-tolerant relational concept learning algorithms. In Birnbaum, L. & Collins, G. (Eds.), *Proc of the 8th Int Workshop on Machine Learning* (pp. 389–393). Evanston, IL: Morgan Kaufmann, San Francisco.

Buntine, W. L. (1992). *A Theory of Learning Classification Rules*. PhD thesis, School of Computing Science, University of Technology, Sydney.

Cendrowska, J. (1987). Prism: an algorithm for inducing modular rules. *Int Journal of Man-Machine Studies*, *27(4)*, 349–370.

Cestnik, B. & Bratko, I. (1991). On estimating probabilities in tree pruning. In Kodratoff, Y. (Ed.), *Proc of the European Working Session on Learning* (pp. 138–150). Porto, Portugal: Springer-Verlag, Berlin.

Cestnik, B., Kononenko, I. & Bratko, I. (1987). ASSISTANT 86: A knowledge-elicitation tool for sophisticated users. In Bratko, I. & Lavrač, N. (Eds.), *Progress in Machine Learning* (pp. 31–45). Bled, Slovenia: Sigma Press, Wilmslow, UK.

Clark, P. & Boswell, R. (1991). Rule induction with CN2: some recent improvements. In Kodratoff, Y. (Ed.), *Proc of the European Working Session on Learning* (pp. 150–163). Porto, Portugal: Springer-Verlag, Berlin.

Clark, P. & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, *3(4)*, 261–283.

Cliff, N. (1987). *Analyzing multivariate data*. San Diego, CA: Harcourt Brace Jovanovich.

Cochran, W. (1954). Some methods of strengthening the common χ^2 tests. *Biometrics*, *10*, 417–451.

Cohen, P. R. & Jensen, D. (1997). Overfitting explained. In *Preliminary Papers of the 6th Int Workshop on Artificial Intelligence and Statistics* (pp. 115–122). unpublished.

Cohen, W. W. (1995). Fast effective rule induction. In Prieditis, A. & Russell, S. (Eds.), *Proc of the 12th Int Conf on Machine Learning* (pp. 115–123). Tahoe City, California: Morgan Kaufmann, San Francisco, CA.

Cohen, W. W. & Singer, Y. (1999). A simple, fast, and effective rule learner. In Hendler, J. & Subramanian, D. (Eds.), *Proc of the 16th National Conf on Artificial Intelligence* (pp. 335–342). Orlando, Florida: AAAI Press, Menlo Park, CA.

Crawford, S. L. (1989). Extensions to the CART algorithm. *International Journal of Man-Machine Studies*, *31(2)*, 197–217.

Dietterich, T. & Kong, E. (1995). Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. Technical report, Dept. of Computer Science, Oregon State University.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, *10(7)*, 1895–1923.

Domingos, P. (1996). Unifying instance-based and rule-based induction. *Machine Learning*, *24(2)*, 141–68.

Efron, B. & Tibshirani, R. (1997). Improvements on cross-validation: The .632+ bootstrap method. *Journal of the American Statistical Association*, *92(438)*, 548–560.

Efron, B. & Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. Chapman and Hall, London, UK.

Esposito, F., Malerba, D. & Semeraro, G. (1995). Simplifying decision trees by pruning and grafting: New results. In Lavrač, N. & Wrobel, S. (Eds.), *Proc of* *the 8th European Conf on Machine Learning* (pp. 287–290). Heraclion, Crete, Greece: Springer-Verlag, Berlin.

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

Fisher, D. (1992). Pessimistic and optimistic induction. Technical Report CS-92-12, Department of Computer Science, Vanderbilt University.

Fisher, D. H. & Schlimmer, J. C. (1988). Concept simplification and prediction accuracy. In Laird, J. (Ed.), *Proc of the 5th Int Conf on Machine Learning* (pp. 22–28). Ann Arbor, MI: Morgan Kaufmann, San Mateo, CA.

Forsyth, R. (1994). Overfitting revisited: an information-theoretic approach to simplifying discrimination trees. *Journal of Experimental & Theoretical Artificial Intelligence*, *6(3)*, 289–302.

Frank, E., Paynter, G. W., Witten, I. H., Gutwin, C. & Nevill-Manning, C. G. (1999). Domain-specific keyphrase extraction. In *Proc of the 16th Int Joint Conf on Artificial Intelligence* (pp. 668–673). Stockholm, Sweden: Morgan Kaufmann, San Francisco, CA.

Frank, E., Trigg, L., Holmes, G. & Witten, I. H. (in press). Naive bayes for regression. *Machine Learning.*

Frank, E., Wang, Y., Inglis, S., Holmes, G. & Witten, I. H. (1998). Using model trees for classification. *Machine Learning*, *32(1)*, 63–76.

Frank, E. & Witten, I. H. (1998a). Generating accurate rule sets without global optimization. In Shavlik, J. (Ed.), *Proc of the 15th Int Conf on Machine Learning* (pp. 144–151). Madison, Wisconsin: Morgan Kaufmann, San Francisco, CA.

Frank, E. & Witten, I. H. (1998b). Using a permutation test for attribute selection in decision trees. In Shavlik, J. (Ed.), *Proc of the 15th Int Conf on Machine Learning* (pp. 152–160). Madison, Wisconsin: Morgan Kaufmann, San Francisco, CA.

Frank, E. & Witten, I. H. (1999). Making better use of global discretization. In Bratko, I. & Dzeroski, S. (Eds.), *Proc of the 16th Int Conf on Machine Learning* (pp. 115–123). Bled, Slovenia: Morgan Kaufmann, San Francisco, CA.

Freund, Y. & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In Saitta, L. (Ed.), *Proc of the 13th Int Conf on Machine Learning* (pp. 148–156). Bari, Italy: Morgan Kaufmann, San Francisco, CA.

Friedman, J. H. (1997). On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, *1(1)*, 55–77.

Fulton, T., Kasif, S. & Salzberg, S. (1995). Efficient algorithms for finding multi-way splits for decision trees. In Prieditis, A. & Russell, S. (Eds.), *Proc of the 12th Int Conf on Machine Learning* (pp. 244–251). Tahoe City, California: Morgan Kaufmann, San Francisco, CA.

Fürnkranz, J. (1997). Pruning algorithms for rule learning. *Machine Learning*, *27(2)*, 139–171.

Fürnkranz, J. (1999). Separate-and-conquer rule learning. *Artificial Intelligence Review*, *13(1)*, 3–54.

Fürnkranz, J. & Widmer, G. (1994). Incremental reduced error pruning. In Cohen, W. W. & Hirsh, H. (Eds.), *Proc of the 11th Int Conf on Machine Learning* (pp. 70–77). Rutgers University, Newark, NJ: Morgan Kaufmann, San Francisco, CA.

Gaines, B. (1989). An ounce of knowledge is worth a ton of data: quantitative studies of the trade-off between expertise and data based on statistically well-founded empirical induction. In Spatz, B. (Ed.), *Proc of the 6th Int Workshop on Machine Learning* (pp. 156–159). Ithaca, NY: Morgan Kaufmann, San Mateo, CA.

Gelfand, S. B., Ravishankar, C. S. & Delp, E. J. (1991). An iterative growing and pruning algorithm for classification tree design. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *13(2)*, 163–174.

Good, P. (1994). *Permutation Tests*. New York: Springer-Verlag.

Goodman, R. & Smyth, P. (1988). Decision tree design from a communication theory standpoint. *IEEE Transactions on Information Theory, 34(5).*

Hall, M., Holmes, G. & Frank, E. (1999). Generating rule sets from model trees. In *Proc of the 12th Australian Joint Conf on Artificial Intelligence* (pp. 1–12). Sydney, Australia: Springer-Verlag, Berlin.

Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, *11(1)*, 63–91.

Holte, R. C., Acker, L. E. & Porter, B. W. (1989). Concept learning and the problem of small disjuncts. In Sridharan, N. S. (Ed.), *Proc of the 11th Int Joint Conf on Artificial Intelligence* (pp. 813–818). Detroit, MI, USA: Morgan Kaufmann, San Mateo, CA.

Hong, S., Hosking, J. & Winograd, S. (1996). Use of randomization to normalize feature merits. In Dowe, D. L., Korb, K. B. & Oliver, J. (Eds.), *Information, Statistics and Induction in Science, Proc of the ISIS 96 Conf* (pp. 10–19). Melbourne, Australia: World Scientific, Singapore.

Indurkhya, N. & Weiss, S. M. (1991). Iterative rule induction methods. *Journal of Applied Intelligence*, *1(1)*, 43–54.

Jensen, D. (1992). *Induction with Randomization Testing.* PhD thesis, Washington University, St. Louis, Missouri.

Jensen, D. & Cohen, P. (2000). Multiple comparisons in induction algorithms. *Machine Learning*, *38(3)*, 1–30.

Jensen, D., Oates, T. & Cohen, P. R. (1997). Building simple models: A case study with decision trees. In Liu, X., Cohen, P. R. & Berthold, M. R. (Eds.), *Advances in Intelligent Data Analysis, Reasoning about Data, Second International Symposium, IDA-97* (pp. 211–222). London, UK: Springer-Verlag, Berlin.

Jensen, D. & Schmill, M. (1997). Adjusting for multiple comparisons in decision tree pruning. In Heckerman, D., Mannila, H., Pregibon, D. & Uthurusamy, R. (Eds.), *Proc of the 3rd Int Conf on Knowledge Discovery and Data Mining* (pp. 195–198). Newport Beach, California: AAAI Press, Menlo Park, CA.

Jordan, M. I. (Ed.). (1999). *Learning in Graphical Models*. MIT Press, Cambridge, MA.

Kalkanis, G. (1993). The application of confidence interval error analysis to the design of decision tree classifiers. *Pattern Recognition Letters*, *14(5)*, 355–361.

Kalles, D. (1995). *Decision trees and domain knowledge in pattern recognition*. PhD thesis, Department of Computation, UMIST.

Kass, G. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, *29(2)*.

Kearns, M. (1996). A bound on the error of cross-validation using the approximation and estimation rates, with consequences for the training-test split. In Touretzky, D. S., Mozer, M. C. & Hasselmo, M. E. (Eds.), *Advances in Neural Information Processing Systems 8* (pp. 183–189). MIT Press, Cambridge, MA.

Kearns, M. & Mansour, Y. (1998). A fast, bottom-up decision tree pruning algorithm with near-optimal generalization. In Shavlik, J. (Ed.), *Proc of the 15th Int Conf on Machine Learning* (pp. 269–277). Madison, Wisconsin: Morgan Kaufmann, San Francisco, CA.

Kim, H. & Koehler, G. J. (1994). An investigation on the conditions of pruning an induced decision tree. *European Journal of Operational Research*, *77*, 82–95.

Knoll, U., Nakhaeizadeh, G. & Tasend, B. (1994). Cost-sensitive pruning of decision trees. In Bergadano, F. & de Raedt, L. (Eds.), *Proc of the 7th European Conf on Machine Learning* (pp. 383–386). Catania, Italy: Springer-Verlag, Berlin.

Kohavi, R. (1995a). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proc of the 14th Int Joint Conf on Artificial Intelligence* (pp. 1137–1143). Montreal, Canada: Morgan Kaufmann, San Francisco, CA.

Kohavi, R. (1995b). *Wrappers for Performance Enhancements and Oblivious Decision Graphs*. PhD thesis, Stanford University, Department of Computer Science.

Kohavi, R. & Sahami, M. (1996). Error-based and entropy-based discretization of continuous features. In Simoudis, E., Han, J. & Fayyad, U. (Eds.), *Proc* *of the 2nd Int Conf on Knowledge Discovery and Data Mining* (pp. 114–119). Portland, OR: AAAI Press, Menlo Park, CA.

Kohavi, R. & Wolpert, D. H. (1996). Bias plus variance decomposition for zero-one loss functions. In Saitta, L. (Ed.), *Proc of the 13th Int Conf on Machine Learning* (pp. 275–283). Bari, Italy: Morgan Kaufmann, San Francisco.

Kononenko, I. (1995). On biases in estimating multi-valued attributes. In *Proc of the 14th Int Joint Conf on Artificial Intelligence* (pp. 1034–1040). Morgan Kaufmann, San Francisco, CA. Montreal, Canada.

Lancaster, H. (1961). Significance tests in discrete distributions. *Journal of the American Statistical Association*, *56*, 223–234.

Li, X. & Dubest, R. C. (1986). Tree classifier design with a permutation statistic. *Pattern Recognition*, *19(3)*, 229–235.

Lock, R. (1991). A sequential approximation to a permutation test. *Communications in statistics: simulation and computation*, *20(1)*, 341–363.

Loh, W.-Y. & Shih, Y.-S. (1997). Split selection methods for classification trees. *Statistica Sinica*, *7*, 815–840.

Loh, W.-Y. & Vanichsetakul, N. (1988). Tree-structured classification via generalized discriminant analysis. *Journal of the American Statistical Association*, *83*, 715–728.

Mansour, Y. (1997). Pessimistic decision tree pruning based on tree size. In Douglas H. Fisher, J. (Ed.), *Proc of the 14th Int Conf on Machine Learning* (pp. 195–201). Nashville, Tennessee: Morgan Kaufmann, San Francisco, CA.

Martin, J. K. (1997). An exact probability metric for decision tree splitting and stopping. *Machine Learning*, *28(2,3)*, 257–291.

Mehta, M., Rissanen, J. & Agrawal, R. (1995). MDL-based decision tree pruning. In Fayyad, U. M. & Uthurusamy, R. (Eds.), *Proc of the 1st Int Conf on Knowledge Discovery and Data Mining* (pp. 216–221). Montreal, Canada: AAAI Press, Menlo Park, CA.

Merriam-Webster (1999). *Merriam-Webster Online Dictionary*. Merriam-Webster.

Michalski, R. & Chilausky, R. (1980). Learning by being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis. *Int Journal of Policy Analysis and Information Systems*, *4(2)*.

Mingers, J. (1987). Expert systems—rule induction with statistical data. *Journal of the Operational Research Society*, *38*, 39–47.

Mingers, J. (1988). An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, *3(4)*, 319–342.

Mingers, J. (1989). An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, *4(2)*, 227–243.

Mitchell, T. M. (1997). *Machine Learning*. London: McGraw-Hill.

Murthy, S. K. (1998). Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, *2(4)*.

Niblett, T. (1987). Constructing decision trees in noisy domains. In Bratko, I. & Lavrač, N. (Eds.), *Progress in Machine Learning* (pp. 67–78). Bled, Slovenia: Sigma Press, Wilmslow, UK.

Niblett, T. & Bratko, I. (1987). Learning decision rules in noisy domains. In Bramer, M. A. (Ed.), *Research and Development in Expert Systems III. Proceedings of Expert Systems '86, Brighton 1986* (pp. 25–34). Cambridge: Cambridge University Press.

Nock, R. & Jappy, P. (1998). On the power of decision lists. In Shavlik, J. (Ed.), *Proc of the 15th Int Conf on Machine Learning* (pp. 413–420). Madison, Wisconsin: Morgan Kaufmann, San Francisco, CA.

Nock, R. & Jappy, P. (1999). Decision tree based induction of decision lists. *Intelligent Data Analysis*, *3(3)*, 227–240.

Oates, T. & Jensen, D. (1997). The effects of training set size on decision tree complexity. In Douglas H. Fisher, J. (Ed.), *Proc of the 14th Int Conf on Ma-*

chine Learning (pp. 254–262). Nashville, Tennessee: Morgan Kaufmann, San Francisco, CA.

Oates, T. & Jensen, D. (1998). Large datasets lead to overly complex models: an explanation and a solution. In Agrawal, R. & Stolorz, P. (Eds.), *Proc of the 4th Int Conf on Knowledge Discovery and Data Mining* (pp. 294–298). New York City, New York: AAAI Press, Menlo Park, CA.

Oates, T. & Jensen, D. (1999). Toward a theoretical understanding of why and when decision tree pruning algorithms fail. In Hendler, J. & Subramanian, D. (Eds.), *Proc of the 16th National Conf on Artificial Intelligence*. Orlando, Florida: AAAI Press, Menlo Park, CA.

Pagallo, G. & Haussler, D. (1990). Boolean feature discovery in empirical learning. *Machine Learning*, *5(1)*, 71–99.

Patefield, W. (1981). An efficient method of generating random $r \times c$ tables with given row and column totals. *Applied Statistics*, *30(1)*, 91–97.

Pereira, F. C. & Singer, Y. (1999). An efficient extension to mixture techniques for prediction and decision trees. *Machine Learning*, *36(3)*, 183–199.

Press, W. H., Teukolsky, S., Vettering, W. & Flannery, B. (1988). *Numerical Recipes in C* (2nd Ed.). Cambridge University Press.

Quinlan, J. (1979). *Expert Systems in the MicroElectronic Age*, chapter Discovering rules by induction from large collections of examples, (pp. 168–201). Edinburgh: Edinburgh University Press.

Quinlan, J. (1986a). *Machine Learning: An Artificial Intelligence Approach*, Volume 2, chapter The effect of noise on concept learning, (pp. 149–166). Los Altos, CA: Morgan Kaufmann.

Quinlan, J. (1987a). Simplifying decision trees. *International Journal of Man-Machine Studies*, *27(3)*, 221–234.

Quinlan, J. (1992). *C4.5: Programs for Machine Learning*. Los Altos, CA: Morgan Kaufmann.

Quinlan, J. (1996). Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research*, *4(1)*, 77–90.

Quinlan, J. & Rivest, R. (1989). Inferring decision trees using the minimum description length principle. *Information and Computation*, *80(3)*, 227–248.

Quinlan, J. R. (1986b). Induction of decision trees. *Machine Learning*, *1(1)*, 81–106.

Quinlan, J. R. (1987b). Generating production rules from decision trees. In McDermott, J. (Ed.), *Proc of the 10th Int Joint Conf on Artificial Intelligence* (pp. 304–307). Milan, Italy: Morgan Kaufmann, Los Altos, CA.

Quinlan, J. R. & Cameron-Jones, R. M. (1995). Oversearching and layered search in empirical learning. In *Proc of the 14th Int Joint Conf on Artificial Intelligence* (pp. 1019–1024). Montreal, Canada: Morgan Kaufmann, San Francisco, CA.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, *14*, 465–471.

Rivest, R. L. (1987). Learning decision lists. *Machine Learning*, *2(3)*, 229–246.

Rumelhart, D. E. & McClelland, J. L. (1986). *Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Vol. 1: Foundations*. Cambridge, MA: MIT Press.

Safavian, S. R. & Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics*, *21(3)*, 660–674.

Salzberg, S. (1991). A nearest hyperrectangle learning method. *Machine Learning*, *6(3)*, 251–276.

Salzberg, S. L. (1997). On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, *1(3)*.

Schaffer, C. (1991). When does overfitting decrease prediction accuracy in induced decision trees and rule sets? In Kodratoff, Y. (Ed.), *Proc of the European Working Session on Learning* (pp. 192–205). Porto, Portugal: Springer-Verlag, Berlin.

Schaffer, C. (1992). Sparse data and the effect of overfitting avoidance in decision tree induction. In Swartout, W. (Ed.), *Proc of the 10th National Conf on*

Artificial Intelligence (pp. 147–152). San Jose, CA: AAAI Press, Menlo Park, CA.

Schaffer, C. (1993). Overfitting avoidance as bias. *Machine Learning*, *10(2)*, 153–178.

Schaffer, C. (1994). A conservation law for generalization performance. In Cohen, W. W. & Hirsh, H. (Eds.), *Proc of the 11th Int Conf on Machine Learning* (pp. 259–265). Rutgers University, Newark, NJ: Morgan Kaufmann, San Francisco, CA.

Schapire, R. E., Freund, Y., Bartlett, P. & Lee, W. S. (1997). Boosting the margin: a new explanation for the effectiveness of voting methods. In Douglas H. Fisher, J. (Ed.), *Proc of the 14th Int Conf on Machine Learning* (pp. 322–330). Nashville, Tennessee: Morgan Kaufmann, San Francisco, CA.

Schapire, R. F. & Singer, Y. (1998). Improved boosting algorithms using confidence-rated predictions. In *Proc of the 11th Annual Conf on Computational Learning Theory* (pp. 80–91). Madison, Wisconsin: ACM Press, New York.

Sethi, I. & Sarvarayudu, G. (1982). Hierarchical classifier design using mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, *4(4)*, 441–445.

Tibshirani, R. (1996). Bias, variance and prediction error for classification rules. Technical report, Dept. of Statistics, University of Toronto.

Ting, K. M. (1994). The problem of small disjuncts: its remedy in decision trees. In Elio, R. (Ed.), *Proc of the 10th Canadian Conf on Artificial Intelligence* (pp. 91–97). Banff, Canada: Morgan Kaufmann, San Mateo, CA.

Ting, K. M. (1998). Inducing cost-sensitive trees via instance weighting. In Żytkow, J. M. & Quafafou, M. (Eds.), *Proc of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery* (pp. 139–147). Trondheim, Norway: Springer-Verlag, Berlin.

Utgoff, P. E., Berkman, N. C. & Clouse, J. A. (1997). Decision tree induction based on efficient tree restructuring. *Machine Learning*, *29(1)*, 5–44.

Vadera, S. & Nechab, S. (1994). ID3, its children and their safety. *BCS Specialist Group on Expert Systems Newsletter*, *31*, 11–21.

Valiant, L. (1984). A theory of the learnable. *Communications of the ACM*, *27(11)*, 1134–1142.

Van den Bosch, A., Weijters, T., Van den Herik, H. J. & Daelemans, W. (1997). When small disjuncts abound, try lazy learning: A case study. In Daelemans, W., Flach, P. & Van den Bosch, A. (Eds.), *Proc Benelearn* (pp. 109–118). Tilburg University, Tilburg.

Wallace, C. S. & Patrick, J. D. (1993). Coding decision trees. *Machine Learning*, *11(1)*, 7–22.

Webb, G. (1994). Recent progress in learning decision lists by prepending inferred rules. In *Proc of the Singapore Int Conf on Intelligent Systems* (pp. 280–285). Singapore: not published.

Weiss, G. M. (1995). Learning with rare cases and small disjuncts. In Prieditis, A. & Russell, S. (Eds.), *Proc of the 12th Int Conf on Machine Learning* (pp. 558–565). Tahoe City, California: Morgan Kaufmann, San Francisco, CA.

Weiss, G. M. & Hirsh, H. (1998). The problem with noise and small disjuncts. In Shavlik, J. (Ed.), *Proc of the 15th Int Conf on Machine Learning* (pp. 574–578). Madison, Wisconsin: Morgan Kaufmann, San Francisco, CA.

Weiss, S. M. & Indurkhya, N. (1994a). Decision tree pruning: Biased or optimal? In *Proc of the 12th National Conf on Artificial Intelligence* (pp. 626–632). Seattle, WA: AAAI Press, Menlo Park, CA.

Weiss, S. M. & Indurkhya, N. (1994b). Small sample decision tree pruning. In Cohen, W. W. & Hirsh, H. (Eds.), *Proc of the 11th Int Conf on Machine Learning* (pp. 335–342). Rutgers University, Newark, NJ: Morgan Kaufmann, San Francisco, CA.

Westfall, P. & Young, S. (1993). *Resampling-based multiple testing: Examples and methods for p-value adjustment*. New York: Wiley.

Wettschereck, D. & Dietterich, T. G. (1995). An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. *Machine Learning*, *19(1)*, 5–27.

White, A. P. & Liu, W. Z. (1994). Bias in information-based measures in decision tree induction. *Machine Learning*, *15(3)*, 321–329.

Wild, C. & Weber, G. (1995). *Introduction to Probability and Statistics*. University of Auckland.

Witten, I. H. & Frank, E. (2000). *Data Mining: Practical Machine Learning Tools and Techniques With Java Implementations*. San Francisco, CA: Morgan Kaufmann.

Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural Computation*, *8(7)*, 1341–1390.

Zheng, Z. (1998). Scaling up the rule generation of C4.5. In Wu, X., Kotagiri, R. & Korb, K. B. (Eds.), *Proc of the 2nd Pacific-Asia Conf on Knowledge Discovery and Data Mining* (pp. 348–359). Melbourne, Australia: Springer-Verlag, Berlin.