THE EFFECT OF SMALL DISJUNCTS AND CLASS DISTRIBUTION ON DECISION TREE LEARNING

by

GARY MITCHELL WEISS

A dissertation submitted to the

Graduate School- New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Computer Science

written under the direction of

Haym Hirsh

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

May, 2003

ABSTRACT OF THE DISSERTATION

The Effect of Small Disjuncts and Class Distribution on Decision Tree Learning

by Gary Mitchell Weiss

Dissertation Director: Haym Hirsh

The main goal of classifier learning is to generate a model that makes few misclassification errors. Given this emphasis on error minimization, it makes sense to try to understand how the induction process gives rise to classifiers that make errors and whether we can identify those parts of the classifier that generate most of the errors. In this thesis we provide the first comprehensive studies of two major sources of classification errors. The first study concerns small disjuncts, which are those disjuncts within a classifier that cover only a few training examples. An analysis of classifiers induced from thirty data sets shows that these small disjuncts are extremely error prone and often account for the majority of all classification errors. Because small disjuncts largely determine classifier performance, we use them as a "lens" through which to study classifier induction. Factors such as pruning, training-set size, noise and class imbalance are each analyzed to determine how they affect small disjuncts and, more generally, classifier learning.

The second study analyzes the effect that rare classes and class distribution have on learning. Those examples belonging to rare classes are shown to be misclassified much more often than common classes. The thesis then goes on to analyze the impact that

varying the class distribution of the training data has on classifier performance. The experimental results indicate that the naturally occurring class distribution is not always best for learning and that a balanced class distribution should be chosen to generate a classifier robust to different misclassification costs. It is often necessary to limit the amount of training data used for learning, due to the costs associated with obtaining and learning from this data. This thesis presents a budget-sensitive progressive-sampling algorithm for selecting training examples in this situation. This algorithm is shown to produce a class distribution that performs quite well for learning (i.e., is near optimal).

Acknowledgements

This dissertation and my doctoral degree could not have been completed without the help, encouragement, and support of many people. It is with great appreciation that I acknowledge these people here.

I would first like to thank my advisor, Haym Hirsh, for his guidance and support. I am especially grateful for the enthusiasm he expressed for my work and the encouragement he provided to me. I would also like to thank Foster Provost, my co-advisor, for the productive working relationship we have had these past few years. I would also like to thank my other committee members, Louis Steinberg and Casimir Kulikowski, for the time and effort they spent in reviewing this dissertation.

I would also like to acknowledge the members of the Rutgers Machine Learning Research group, for their many helpful comments as well as for their support and friendship: Arunava Banerjee, Chumki Basu, Brian Davison, Daniel Kudenko, David Lowenstern, Sofus Macskassy, Chris Mesterharm, Khaled Rashed, and Sarah Zelikovitz. I would also like to thank the departmental graduate secretary through most of my years at Rutgers, Valentine Rolfe, for always showing a personal interest in me as well as other students.

I would like to thank my employer, AT&T Laboratories, and especially the various managers I have had over the years who have supported my academic pursuits; in these times such support is indeed quite rare. I would like to acknowledge the support of these managers by name: John Palframan, Ed Moke, Raj Dube, Jennifer Thien, Dave Trutt and Tom Au.

I would like to thank William Cohen for supplying the AT&T data sets used throughout this dissertation and for providing detailed information about the Ripper rule-induction program.

Finally, I would like to thank all of my family. I want to thank my wife, Rosalie, for her patience these past several years and especially for encouraging me to enter the doctoral program in the first place. I'd also like to thank my son, Jaden, for being such a wonderful boy and for providing me a great incentive for completing my degree.

Dedication

To Rosalie, who motivated me to start,

and Jaden, who motivated me to finish.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

"Errors to be dangerous must have a great deal of truth mingled with them"

- Sydney Smith

A general goal of classifier learning is to generate a model, based on training data, which makes as few errors as possible when classifying new, previously unseen, examples. Given this emphasis on minimizing the number of classification errors, it makes sense to try to understand how the induction process gives rise to classifiers that make errors and whether these errors have any structure to them. That is, can we identify the portions of a classifier, or aspects of a learning problem, that lead to most of the classification errors? In this thesis we demonstrate that the answer is "yes". Because it is difficult to study classifier induction in general, in this thesis we focus our attention primarily on one popular class of learners—decision tree learners (some results are also presented for rule learners).

Two major sources of classification errors are identified and studied in depth. The first source of errors are those learned classification rules that are generated from relatively few training examples (called small disjuncts). These rules are shown, in many cases, to contribute most of the classification errors, even though they classify relatively few examples. The second source of errors are rare, or underrepresented, classes. Domains with rare classes prove to be difficult for induction systems. While this problem is

associated with a characteristic of the learning problem, there is a structure to the errors. In particular, classification rules predicting the rare class tend to be very error prone and most classification errors occur when classifying examples belonging to the rare class. The first source of errors is studied in Chapters 3 - 5 while the second source of errors is studied in Chapters 6 - 9.

These two sources of errors are investigated empirically, by analyzing the classifiers induced from a large set of benchmark data sets. Each of these two studies begins by measuring, or quantifying, the extent to which these sources are responsible for misclassification errors. The results confirm that these sources of errors really do account, in many cases, for the majority of all errors. Each of the two studies then analyzes the results in order to determine the structure of the problem and then proceeds to explain the phenomena, drawing on some basic theory.

Each study then conducts additional experiments in order to better understand the source of the errors are how various factors contribute to these errors. For the first study, factors such as the pruning strategy, amount of noise, training-set size and class imbalance are varied in order see how these factors affect the distribution of errors in the induced classifier, and, in particular, how they affect the classification rules that are generated from few training examples. These results provide insight into how each of these important factors affects inductive learning. The second study varies the class distribution of the training data. These experiments show how changing the class distribution of the training data affects the ability of the learner to classify examples belonging to the rare and common classes, and, more importantly, how this affects overall classifier per-

formance. This enables us to identify and characterize the class distribution that is best for learning.

This chapter is organized as follows. It begins by providing a brief overview of classifier induction and decision tree induction, focusing on those aspects of learning that are essential to the thesis. The motivations for studying small disjuncts and class distribution are then discussed. This is followed by a summary of the main contributions of the thesis. An outline of the remainder of the thesis is then presented.

1.1 Overview of Classifier Induction

Inductive learning involves forming generalizations from specific examples. Classifier induction, also referred to as classifier learning, is a type of inductive learning that takes a set of labeled training examples and induces a classifier that can classify unlabeled training examples. Classifier induction plays a central role in machine learning and has many practical and diverse applications—it can be used to identify fraudulent transactions (Fawcett & Provost, 1997), predict telecommunication faults (Weiss & Hirsh, 1998) and automatically categorize news articles (Glover, Tsioutsiouliklis, Lawrence, Pennock & Flake, 2002).

The goal of classifier induction is to learn the true, or *target*, concept that underlies the data. Except for artificially generated data sets, the target concept is generally not known. A simple example will be used to demonstrate the basics of classifier induction. This example has to do with classifying an animal into one of two classes: *bird* or *not-a-bird*. In this example, the target concept *bird* might be defined by an ornithologist and hence be known. This target concept could be represented in many ways. A rule-based representation is used to represent the target concept in Figure 1.1. The rules are executed in order

and classification is made based on the first rule for which the left side evaluates to true. The third rule is a default rule that classifies the animal as *not-bird*.

1) flies(x) ∧ feathers(x) ∧ has-beak(x) ∧ lays-eggs(x) => bird(x)

2) wings(x) ∧ ⌐flies(x) ∧ has-beak(x) ∧ lays-eggs(x) => bird(x)

3) not-bird(x)

Figure 1.1: Target Concept for bird

One might be given a classified list of animals, with descriptions, and asked to generate a classifier for the concept *bird* based only on these training examples. Such a list of examples is provided in Table 1.1, where each column represents an attribute that describes the animal. Note that in this case the animals are represented using attribute-value pairs rather than predicates, but for our purposes these two methods are equivalent. Note, however, that several additional pieces of information are introduced in Table 1.1: name, milk (i.e., produces milk), hair/fur, and air (i.e., breathes air). The last column specifies the classification. In this case the class value is not restricted to *bird* or *not-bird*, but we will assume that all values not equal to *bird* will be replaced with *not-bird*.

name	**flies**	**milk**	**lays-eggs**	**hair/fur**	**air**	**feathers**	**wings**	**class**
pigeon	yes	no	yes	no	yes	yes	yes	bird
wasp	yes	no	yes	no	yes	no	yes	insect
lion	no	yes	no	yes	yes	no	no	mammal
eagle	yes	no	yes	no	yes	yes	yes	bird
catfish	no	no	yes	no	no	no	no	fish
cardinal	yes	no	yes	no	yes	yes	yes	bird
ostrich	no	no	yes	no	yes	yes	yes	bird

Table 1.1: Training data for Concept “Bird”

The training data described in Table 1.1 might lead to the classifier in Figure 1.2. This classifier is represented using a decision tree. Examples are labeled by starting at the root node of the decision tree and proceeding to a leaf, which contains the class label. The path that is taken is determined by evaluating the conditions at the internal nodes, using the data associated with the example. This classifier correctly classifies all of the training examples in Table 1.1. Note that there are many possible decision trees that could be formed from the data in Table 1.1 and that the classifier shown in Figure 1.2 is by no means the simplest, or best. In fact, the entire right sub-branch could be pruned, since all animals that do not breathe air belong to *not-bird*.

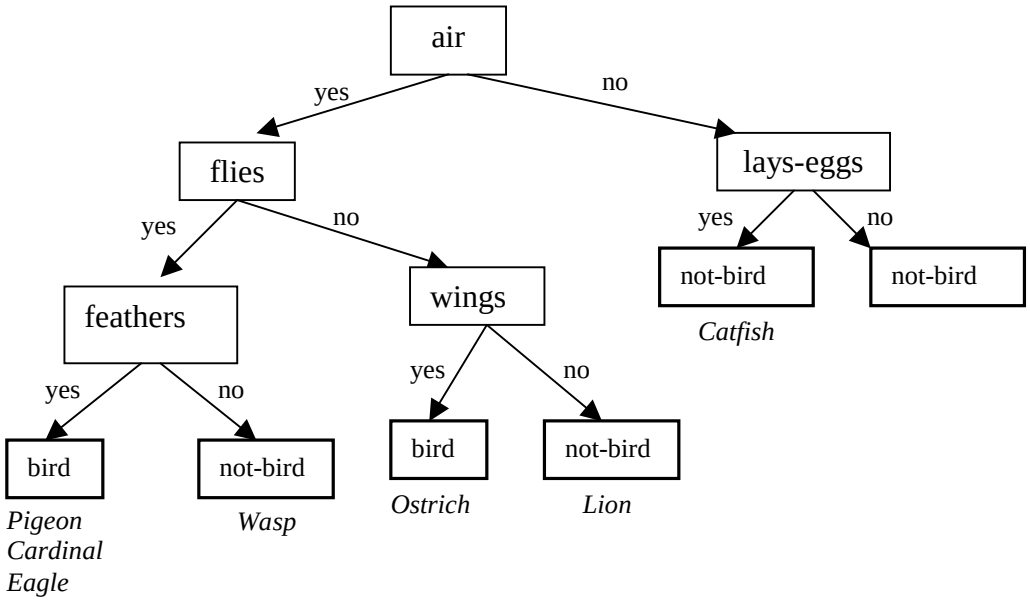

Figure 1.2: Classifier for Concept “Bird”

In general, the learned concept and the target concept are not the same—the learned concept is only an approximation of the target concept. To see this for the example just described, note that if the presence or absence of a beak were necessary to distinguish between *bird* and *not-bird*, then the classifier would yield an incorrect classification.

One way to categorize classifier induction systems is by the structure, or language, used to represent the induced classifier. The above example uses a decision tree, but other common learners represent the learned classifier using sets of classification rules, a neural network, or, in the case of instance-based learning, a set of labeled examples with an associated similarity metric.

Many classifiers operate by explicitly partitioning the instance space into regions and associating a class with that region. Thus, any examples that fall into that region are assigned the designated class. Decision tree learners and rule learners are two types of learners that generate classifiers that operate in this manner. For a given class, each region associated with that class can be considered a subconcept of the concept associated with that class. For example, the concept *bird* in Figure 1.2 can be expressed as:

$$bird = (\text{air:yes} \wedge \text{flies:yes} \wedge \text{feathers:yes}) \vee (\text{air:yes} \wedge \text{flies:no} \wedge \text{wings:yes})$$

This concept is formed from two subconcepts, which could be assigned the descriptive names *flying-bird* and *grounded-bird*, respectively, since the first represents birds that fly and the second one represents birds that do not fly. Each of these subconcepts may be viewed as a disjunct. Similarly, each leaf in a decision tree may be considered a disjunct. Throughout this thesis we use the term disjunct exclusively to refer to subconcepts defined by the *learned* classifier, not the target concept.

One can talk about the *size* of a subconcept based on the number, or proportion, of examples that is covers, given some distribution of examples, *D*. We call subconcepts that will cover many examples (i.e., define a region in the instance space that covers many examples from the distribution *D*) a large subconcept while we call subconcepts that will cover relatively few examples a small subconcept. One subconcept is smaller than an-

other if, based on *D*, it is expected to cover fewer examples. Note that the size of a subconcept is determined not only by the size of the region it covers, but also by *D*.

Holte, Acker & Porter (1989) define the *size* of a disjunct as the number of training examples that the disjunct correctly classifies. Given the discussion of the size of a subconcept, there are two important things to note. First, the size of a disjunct is defined in terms of the training data, rather than the distribution *D*. This is required because *D* is typically not known. Secondly, the size of a disjunct is defined in terms of the number of correctly classified training examples, not the total number of training examples classified. This definition of disjunct size is used in previous research on small disjuncts and is used throughout this thesis.

Applying this definition of disjunct size to the *bird* example, we see that the disjunct representing *flying-bird* in Figure 1.2 has a disjunct size of three, since it correctly classifies pigeon, cardinal, and eagle, while the disjunct representing *grounded-bird* has a disjunct size of one, since it only classifies ostrich.

Previous research has shown that large subconcepts in a target concept will often result in large disjuncts in the learned classifier, just as small subconcepts in the target concept will often result in small disjuncts in the learned classifier (Weiss, 1995). However, the correspondence is not normally as clear as it is in the simple example presented. As we show in Chapter 3, the bias and expressive power of a learner affects the formation of small disjuncts. Previous research (Weiss, 1995; Weiss & Hirsh, 1998) also suggests that other factors, such as noise and training-set size may affect the formation of small disjuncts.

Every data set, or domain, has an associated class distribution, which measures the proportion of examples belonging to each class. In the training data in Table 1.1, the class distribution is 4 examples of class *bird* to 3 examples of class *not-bird*. In this case, the classes are relatively well balanced. However, in many real-world data sets classes are often highly unbalanced. In such a situation classes we refer to classes that are severely underrepresented as *rare classes*.

We can also talk about the class distribution of a domain, rather than the distribution of a specific data set. In the bird example, based on other information we might know that the underlying, or naturally occurring, class distribution is actually 1:1 instead of 4:3. If this were the case, the training data would not be representative of the true, underlying, distribution. Given that most induction algorithms assume by default that the training data mimics the naturally occurring distribution, this will cause a problem. As we show later on, such differences in distribution must be taken into account.

1.2 Motivation

There are a variety of reasons for studying small disjuncts and class distribution. While some of these reasons were briefly mentioned toward the beginning of this chapter, the benefits of studying small disjuncts and class distribution are more fully discussed in this section.

1.2.1 Why Study Small Disjuncts?

A number of studies (Holte, Acker & Porter, 1989; Ali & Pazzani, 1992; Danyluk & Provost, 1993; Weiss, 1995; Weiss & Hirsh, 1998; Carvalho & Freitas, 2000) have shown that small disjuncts have a much higher error rate than large disjuncts, cover a significant

portion of the test examples and contribute a significant, and disproportionate, percentage of all test errors. This phenomenon has been referred to as "the problem with small disjuncts". Therefore, as discussed earlier, one reason to study small disjuncts is because they largely determine—and limit—the performance of the induced classifier.

A second, but related reason for studying small disjuncts is to provide a better understanding of how they affect learning. Factors such as pruning, training-set size, noise and class distribution are each analyzed to see *how* they impact small disjuncts, and, more generally, the distribution of errors with respect to disjunct size. The reasons for studying how these factors affect small disjuncts is not only to better understand small disjuncts and their role in learning, but to gain insight into each of these factors and how they affect learning. Thus, in this thesis small disjuncts are used as a lens through which to view machine learning.

The final motivation for studying small disjuncts is to learn how to address the problem with small disjuncts. That is, can a better understanding of small disjuncts lead to a method for reducing the error rate of the examples that would normally be classified by the small disjuncts? This better understanding appears to be necessary since past efforts to address the problem with small disjuncts, which are described in Chapter 5, have largely failed.

1.2.2 Why Study Class Distribution?

There are two major reasons for studying the effect that the class distribution of the training data has on classifier learning. The first reason is to provide a better understanding of the role of class distribution in learning. The improved understanding provided in this

thesis will permit us to answer the following important questions, many of which are of direct, practical, importance:

- How does class imbalance affect learning?
- Does class imbalance make learning more difficult or easier?
- Why are minority class examples harder to classify than majority class examples?
- What are the implications of changing the class distribution of the training data? Also, how does one account for alterations made to the class distribution of the training data?
- What distribution is best for learning? Can we characterize such an "optimal" distribution? If so, can we provide useful guidelines?

The second motivation for studying class distribution concerns how to choose the class distribution of the training data, when the amount of training data must be limited due to costs associated with procuring the data or learning from the data. In this case, the questions addressed in this thesis are:

- When the amount of training data must be limited to *n* examples, what is the best class distribution for learning?
- Given that only *n* training examples are permitted, how does one go about choosing *n* examples such that the resulting class distribution performs well (i.e., near optimally)?

1.3 Contributions

This thesis makes several major contributions. First, existing studies on small disjuncts and class distribution and their effect on classifier learning has been ad-hoc and has considered only a few data sets. Because of this, general conclusions were not possible and the impact of small disjuncts and class distribution on learning, in general, could not be assessed. Thus, one contribution of this thesis is that it provides a thorough empirical study of small disjuncts and class distribution that analyzes a large number of data sets in a principled manner. This makes it possible, for the first time, to *quantify* the impact that small disjuncts and class distribution have on learning. For small disjuncts, this analysis is greatly facilitated by the introduction of a new metric, error concentration, which makes it feasible, for the first time, to compare the distribution of errors (by disjunct size) across classifiers.

Another contribution of this thesis is that it provides an *understanding* of small disjuncts and class distribution and how they affect learning. This understanding includes explanations for why small disjuncts are more error prone than large disjuncts, why minority-class predictions are more error prone than majority-class predictions, and why minority-class test examples are misclassified more often than their majority-class counterparts.

This thesis also uses small disjuncts to provide a better understanding of decision-tree learning. In particular, the focus on small disjuncts provides insight into how pruning, noise, training-set size and class imbalance affect decision-tree learning. Some of these insights yield practical suggestions of how to improve learning.

This thesis also discusses the impact of changing the class distribution of the training data and shows that such changes will improperly bias the learner and seriously degrade classifier performance, unless the change in distribution is explicitly taken into account. This is quite significant because previous research on the effect of class distribution on learning (Catlett, 1991; Chan & Stolfo, 1998; Japkowicz, 2002) has not accounted for these changes. Consequently this thesis addresses a serious methodological issue with previous research.

Finally, this thesis addresses the issue of data cost and shows that when the amount of training data must be limited, the impact on classifier performance can be minimized by carefully selecting the class distribution of the reduced training set. A budget-sensitive progressive-sampling strategy is described that intelligently selects training examples based on their class. This sampling algorithm is shown to form a class distribution that is nearly optimal for learning. Thus, this thesis makes a major contribution by showing how to select training data when training examples are costly.

1.4 Dissertation Outline

The remainder of this dissertation is organized as follows. First, in Chapter 2, we describe the induction programs, metrics for evaluating classifier performance, and data sets that are used throughout this dissertation. The remainder of the dissertation is then organized into two main parts. Chapters 3 through 5 are dedicated to small disjuncts while Chapters 6 through 9 are dedicated to the effect of class distribution, and costly training data, on learning.

Chapter 3 introduces the problem associated with small disjuncts and then quantifies the problem with small disjuncts by analyzing the classifiers induced from thirty data

sets. Chapter 4 investigates how pruning, training-set size, noise and class imbalance affect small disjuncts, and, more generally, the distribution of errors with respect to disjunct size. Chapter 5 described previous research on small disjuncts.

Chapter 6 introduces the issues related to the study of class distribution and analyzes the classifiers induced from twenty-six large data sets, with respect to differences in performance between the minority class and majority class. Explanations for the observed differences are also provided. Chapter 7 then shows how varying the class distribution of the training data affects classifier performance and analyzes the relationship between training-set size, class distribution and classifier performance. Chapter 8 describes and evaluates a budget-sensitive progressive-sampling algorithm for intelligently selecting training data. Chapter 9 described research related to the role of class distribution on learning.

Chapter 10 concludes the thesis. It summarizes the main conclusions and contributions of the thesis, describes limitations with this research and suggests several avenues for future research.

Chapter 2

Background

Don't be too timid and squeamish about
your actions. All life is an experiment.
The more experiments you make the better.

- Ralph Waldo Emerson

This chapter provides background information that is assumed throughout the remainder of the thesis. In particular, this chapter introduces the classifier induction programs employed in this thesis, describes how classifier performance is to be evaluated, and provides a brief description of the data sets used to study the effect of small disjuncts and class distribution on decision-tree learning.

2.1 Classifier Induction Programs

A classifier induction program takes as input a set of labeled training examples and generates a classifier capable of classifying unlabeled examples. Two such programs are employed in this thesis: C4.5 (Quinlan, 1993) and Ripper (Cohen, 1995). Both induction programs employ pruning to reduce the complexity of the classifier and to improve predictive accuracy. These pruning strategies will be studied to see how they affect learning, and, in particular, how they affect the formation of small disjuncts.

The analysis of small disjuncts and class distribution provided in this thesis requires that the learners record some basic descriptive statistics related to disjunct size and class

distribution. Software scripts then use these descriptive statistics to generate higher-level summary statistics. These summary statistics describe the distribution of errors with respect to disjunct size, the accuracy of minority-class and majority-class predictions, and the accuracy with which the classifier can classify minority-class and majority-class test examples. Both C4.5 and Ripper provide the necessary descriptive statistics for class distribution, but only Ripper supplies sufficient information to track classifier performance with respect to disjunct size. Consequently the C4.5 source code was modified to provide the necessary disjunct-related descriptive statistics.

Other than this one innocuous change to C4.5, which does not affect the classifier that is generated, no changes were made to the source code of either learner. However, as described later, in some cases each learner's pruning strategy is disabled. Also, as described in Chapter 6, for the class distribution experiments the generated classifiers are often modified after the induction programs complete, to take into account changes made to the class distribution of the training data.

2.1.1 C4.5

C4.5 is a popular classifier induction program for learning decision trees (Quinlan, 1993). Given a set of examples described by a fixed number of pre-defined attributes and a class variable with a fixed number of values, C4.5 generates a decision tree capable of assigning one of these class values to each example. The algorithm for generating the tree is described as follows. At every step in the tree-building process, if the remaining examples are all of the same class, a terminal leaf is generated and labeled with that class. Otherwise, the current tree node is split based on the attribute with the highest information gain (Quinlan, 1993). Information gain is an entropy-based metric that measures the

change in impurity of examples in partition E versus examples in partition E_1, E_2, …, E_n, where E_1, E_2, …, E_n is a partition of E. In the case of a decision tree, partition E represents a node in the decision tree and E_1, E_2, …, E_n represent the children nodes. Informally speaking, information gain is used to determine the attribute that will yield a partition of E that yields the most "pure" sub-partitions (i.e., child leaves). Ideally, an attribute will be chosen that yields child nodes that contain homogeneous examples—examples that all belong to a single class. Once the attribute to split on is chosen, the examples are split into one subset per discrete value, or into two subsets for continuous attributes. This procedure continues recursively.

Once the decision tree is generated, C4.5 prunes the tree to avoid overfitting, using Quinlan's pessimistic error pruning (Quinlan, 1987). This pruning strategy works as follows. For a node *n*, all examples covered by *n* (passed down from the root node to *n*) are used to estimate the error rate if *n* were a leaf node, labeled with the majority class at *n*. The estimated error rate assumes a binomial distribution and with 75% probability is an upper bound on the true error. This upper bound, a pessimistic estimate of the true error, is then used in the pruning process. C4.5 prunes the tree bottom-up considering two types of pruning. Specifically, for each node *n*, C4.5 computes 1) the total estimated error of all of *n*'s children, 2) the estimated error if *n* is made a leaf, and 3) the estimated error if *n* is replaced by the branch that covers the most examples. The alternative with the lowest estimated error is chosen and the tree is modified accordingly, such that 1) the tree remains unchanged, 2) the node *n* is pruned to become a leaf, or 3) the node *n* is replaced by the largest branch (known as sub-tree raising).

The C4.5 source code was modified to provide statistics related to disjunct size. During the training phase each disjunct/leaf is assigned a disjunct size based on the number of training examples it correctly classifies. During the testing phase the number of correctly and incorrectly classified examples associated with each disjunct is recorded. Then, when the program terminates, the distribution of correctly and incorrectly classified test examples is printed out, for each disjunct size. For example, the enhanced version of C4.5 might record the fact that disjuncts of size three collectively classify five test examples correctly and three test examples incorrectly.

2.1.2 Ripper

Ripper is a program for inducing sets of classification rules (Cohen, 1995). Each rule is a conjunction of conditions on attribute values. Rules are returned as an ordered list and the first rule that evaluates to true is used to assign the classification. One distinguishing characteristic of Ripper is that it allows attributes to take on sets as values, in addition to nominal and continuous values.

Ripper forms rules as follows: it splits uncovered examples (i.e., those not currently covered by a rule) into a growing set and a pruning set. Rules are grown by adding conditions one at a time until the rule covers only a single example in the growing set. Assuming that *PrunePos* refers to the set of positive examples in the pruning set and *PruneNeg* to the set of negative examples in the pruning set, the rule is then immediately pruned by deleting any final sequence of conditions in the rule that maximize the function:

$$v^*(Rule, PrunePos, PruneNeg) = \frac{p-n}{p+n},$$

where *p* is the number of examples covered by *Rule* in *PrunePos* and *n* is the number of examples covered by *Rule* in *PruneNeg*. Once a rule is created, the examples covered by it are removed from the training data. Ripper continues to generate rules until its stopping criterion is met. The stopping criterion is based on calculating the description length (Barron, Rissanen, & Yu, 1998) of the rule set and stopping if this is more than *d* bits larger than the smallest description length found thus far.

Ripper then performs an optimization step. Considering each rule in the rule set in the order they were generated, Ripper considers two alternatives. The first alternative is replacing the rule by growing and pruning a new rule from scratch, where pruning is done to minimize the error rate of the overall rule set. The second alternative involves revising the existing rule by growing it (i.e., starting with the rule rather than the empty rule) and then pruning it back. In this case pruning uses minimum description length to decide whether to use the original rule, replacement rule, or revised rule. Finally, rules are added to cover positive examples not covered by the optimized rule set. Ripper then repeats this optimization step one more time, since this has been shown to improve classifier performance.

Ripper as described is for a two-class learning problem. Multi-class learning problems are normally handled as follows. First, classes are ordered in terms of increasing levels of prevalence, such that in the sequence C_1, C_2, …, C_k, C_1 is the least frequently occurring class and C_k is the most frequently occurring class. Ripper first finds a rule set to separate C_1 from the remaining classes. Then, all of the examples covered by the rule set are removed from the training data and Ripper is used to separate C_2 from the remaining

classes, etc. This process is repeated until only the last class, C_k, is left. A default rule is used to cover this entire class.

Ripper, using the "-a *unordered*" option, can also be used to produce a set of rules where the classification is not depended on the order of the rules. In this case Ripper will separate each class from the previous class such that a non-overlapping set of rules is generated for each class. Conflicts between classes are resolved by selecting the class with the lowest accuracy on the training data.

2.2 Evaluating Classifier Performance

Empirical studies of classifier induction, such as the ones described in this thesis, require a means of measuring, or evaluating, classifier performance. In some situations, such as when deciding which class distribution yields the best performing classifier, it is important to evaluate the *overall* performance of the classifier. In this thesis overall classifier performance is measured using accuracy, described in Section 2.2.2, and the area under the ROC curve, which is described in Section 2.2.3. In other situations it may be necessary to evaluate some specific aspect of classifier performance, such as the performance of a specific disjunct or of minority-class predictions. In all cases it is essential to have an appropriate evaluation metric—the choice of evaluation metric will affect the results and the conclusions drawn from these results.

In order to analyze exactly how class distribution affects learning, it is necessary to measure classifier performance with respect to each class. In order to simplify the presentation and the analysis of results, only two-class learning problems are studied in this thesis. The less frequently occurring class is referred to as the minority class while the more frequently occurring class is referred to as the majority class. Because machine

learning generally uses the term positive class and negative class for two class problems, in some circumstances we use these terms instead of minority class and majority class. However, throughout this thesis the positive class will always correspond to the minority class and the negative class to the majority class. These associations reflect the tendency to refer to the positive class as the primary class of interest; in this research we consider the minority class to be the more interesting class. This association between the positive class and minority class also reflects how these terms are most often applied in practice. For example, in medical diagnosis "a positive" generally reflects the presence of a disease, which typically is detected less frequently than the absence of a disease.

So, to understand the role of class distribution in learning requires that we track—and compare and contrast—the performance of minority-class and majority-class predictions, as well as the ability of a classifier to correctly classify minority-class and majority-class test examples. The evaluation metrics for doing this are described in Section 2.2.1.

We also use an additional metric specifically designed to measure the impact of small disjuncts on learning. This metric, error concentration, measures the distribution of errors with respect to disjunct size. Because this metric is new and is a contribution in itself, its description is deferred until Chapter 3, where we discuss small disjuncts in detail.

2.2.1 Basic Metrics for Measuring Performance for Two-Class Problems

The basic building blocks for measuring classifier performance for two-class problems are defined in the confusion matrix, shown in Table 2.1. A classifier t is a mapping from instances x to classes {p, n} and is an approximation of c, which represents the true, but unknown, classification function. For notational convenience, let $t(x) \in \{P, N\}$ and $c(x) \in \{p, n\}$ so that it is always clear whether a class value is an actual value (lower case) or

predicted (upper case) value. A true positive (true negative) refers to the case where a positive (negative) example is classified correctly. A false positive corresponds to the case where a negative example is misclassified as a positive example while a false negative corresponds to the case where a positive example is misclassified as a negative example.

		$t(x)$	
		Positive Prediction	Negative Prediction
$c(x)$	Actual Positive	*tp* (true positive)	*fn* (false negative)
	Actual Negative	*fp* (false positive)	*tn* (true negative)

Table 2.1: Confusion Matrix for a Two-Class Problem

The terms defined in Table 2.1 are used to define the higher-level performance metrics in Table 2.2. The metrics described in the first two rows of Table 2.2 measure the ability of a classifier to classify positive and negative *examples* correctly, while the metrics described in the last two rows of the table measure the effectiveness of the *predictions* made by a classifier.

$TP = Pr(P\|p) \approx \frac{tp}{tp+fn}$	*True Positive Rate (recall or sensitivity)*	$FN = Pr(N\|p) \approx \frac{fn}{tp+fn}$	*False Negative Rate*
$TN = Pr(N\|n) \approx \frac{tn}{tn+fp}$	*True Negative Rate (specificity)*	$FP = Pr(P\|n) \approx \frac{fp}{tn+fp}$	*False Positive Rate*
$PPV = Pr(p\|P) \approx \frac{tp}{tp+fp}$	*Positive Predictive Value (precision)*	$\overline{PPV} = Pr(n\|P) \approx \frac{fp}{tp+fp}$	
$NPV = Pr(n\|N) \approx \frac{tn}{tn+fn}$	*Negative Predictive Value*	$\overline{NPV} = Pr(y\|N) \approx \frac{fn}{tn+fn}$	

Table 2.2: Classifier Performance Metrics for Two-Class Problems

The true positive rate, TP, or recall, is the likelihood that a positive example is classified correctly, while the true negative rate, TN, or specificity, is the likelihood that a

negative example is classified correctly. The positive predictive value (PPV), or precision, of a classifier is the likelihood that a positive prediction is correct, while the negative predictive value, NPV, is the likelihood that a negative prediction is correct. Finally, the metrics in the second column of Table 2.2 are the complements of the corresponding metrics in the first column, and can alternatively be computed by subtracting the value in the first column from 1. For example, the false negative rate, FN, is the likelihood that a positive example is classified as a negative example, or, alternatively, is 1 – TP. Because error rate is defined as 1 – accuracy, FN can also be interpreted as the error rate associated with the positive examples. Note that in two cases we are aware of no generally accepted terms for the metrics—these metrics will be referred to using the complement of the names that appear in the first column (e.g., $\overline{PPV}$ and $\overline{NPV}$).

The metrics in Table 2.2 can be summarized succinctly as follows. Proceeding from row 1 through 4, the metrics in column 1 (column 2) represent:

1) The accuracy (error rate) when classifying positive/minority examples
2) The accuracy (error rate) when classifying negative/minority examples
3) The accuracy (error rate) of the positive/minority predictions
4) The accuracy (error rate) of the negative/majority predictions.

When describing the experimental results in Chapter 6 we use the terms in column 2 rather than those in column 1, since it is more revealing to compare error rates than accuracies. For example, an increase in error rate from 1% to 2% corresponds to a relative increase in error rate of 100% but a relative decrease in accuracy of only about 1% (i.e., from 99% to 98%).

2.2.2 Accuracy

Classification accuracy is often used to measure the overall performance of a classifier. It is defined as the fraction of examples classified correctly, or, using the terms from the confusion matrix in Table 2.1, as $(tp + fp)/(tp + fp + tn + fn)$. Equivalently, classifier performance can be specified in terms of error rate, which is defined as 1 – accuracy. The benefits of using accuracy to measure classifier performance are that it is simple to calculate and understand and is the most common evaluation metric in machine-learning research.

Unfortunately there are a number of problems with accuracy—problems that are getting more attention now that research is focusing on increasingly complex, real world, problems. First, accuracy assumes that the target (marginal) class distribution is known and unchanging and, more importantly, that the error costs—the costs of a false positive and false negative—are equal. These assumptions are unrealistic in many domains (Provost et al. 1998). Accuracy is particularly suspect as a performance measure when studying the effect of class distribution on learning since, as we discuss in Chapter 3, it is heavily biased to favor the majority class. Furthermore, highly unbalanced data sets typically have highly non-uniform error costs that favor the minority class, which, as in the case of medical diagnosis and fraud detection, is the class of primary interest. Classifiers that optimize for accuracy for these problems are of questionable value since they rarely predict the minority class.

For the reasons just listed, we do not rely solely on accuracy to measure classifier performance. Instead, classifier performance is also measured using ROC analysis, which addresses the problems with accuracy. Accuracy-related results are included so that this

research can be related to past and future research that may only utilize accuracy—and to highlight the differences that arise when using these two different performance metrics.

2.2.3 ROC Analysis

Receiver Operating Characteristic (ROC) analysis is a method for evaluating classifier performance (Swets et al., 2000). ROC analysis represents the false-positive rate (FP) on the x-axis of a graph and the true-positive rate (TP) on the y-axis (these terms are defined in Table 2.2). Many classifiers, including C4.5, produce not only a classification but also a probability estimate of class membership. ROC curves are produced by varying the threshold on the class-probability estimates (described in detail in Chapter 6). For example, for a decision-tree learner one point on an ROC curve may correspond to the case where the leaves of a decision tree are labeled with the minority class only if the probability of an example at a leaf belonging to the minority class is greater than .5; another point on the curve may correspond to a probability threshold of .1. The use of ROC analysis for machine learning is described in detail elsewhere (Bradley, 1997; Provost & Fawcett, 2001). One advantage of ROC analysis over accuracy is that it evaluates the performance of a classifier independent of the class distribution of the test set or of the error costs.

Several sample ROC curves are shown in Figure 2.1. In this particular example, which is analyzed in detail in Chapter 7, each of the four curves correspond to classifiers induced from the same data set, but using different class distributions for training. In ROC space, the point (0,0) corresponds to the strategy of never making a positive/minority prediction and the point (1,1) to always predicting the positive/minority class. Points to the “northwest” indicate improved performance.

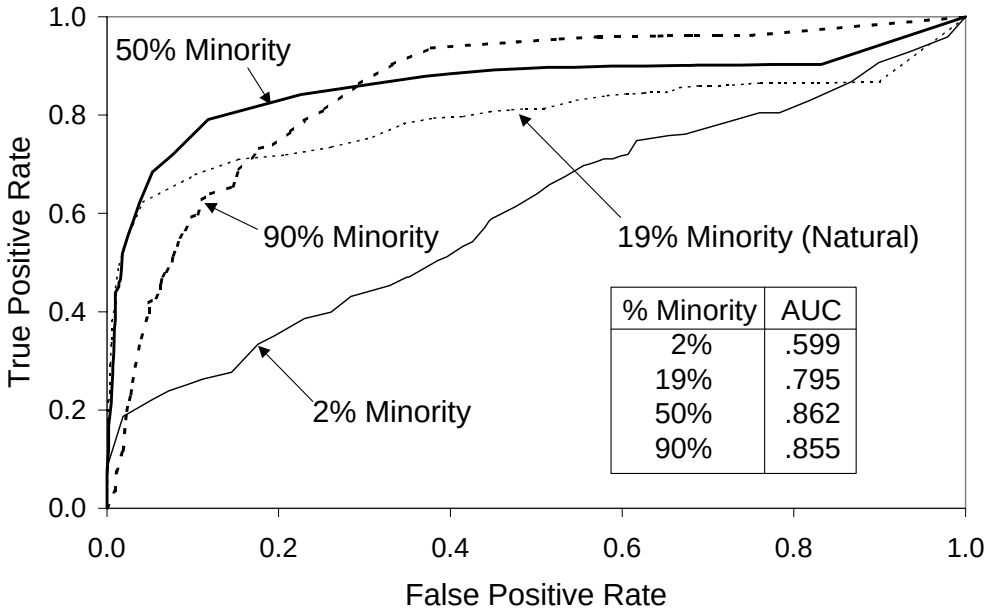

Figure 2.1: Sample ROC Curves

Figure 2.1 shows that different ROC curves may perform better in different areas of ROC space (for this example this indicates that one class distribution does not always lead to the best performance). To assess the *overall* quality of a classifier we measure AUC, the fraction of the total area that falls under the ROC curve, which is equivalent to several statistical measures for evaluating classification and ranking models (Hand, 1997). Larger AUC values indicate generally better classifier performance and, in particular, indicate a better ability to rank cases by likelihood of class membership. It should be kept in mind that if one ROC curve does not dominate the rest, then for *specific* cost and class distributions the best model may not be the one that maximizes AUC. If there is not a single dominating ROC curve, as the case in Figure 2.1, then multiple classifiers can be combined to form a classifier that performs optimally for all costs and distributions (Provost & Fawcett, 2001).

2.3 Data Sets

This thesis provides an empirical study of small disjuncts and class distribution. So that valid conclusions may be drawn, a large number of data sets are analyzed. This section describes the data sets used throughout this thesis. Because, as described shortly, the study of class distribution requires larger data sets than the study of small disjuncts, two different, but overlapping, groups of data sets are described.

The thirty data sets used to study small disjuncts are described in Table 2.3. The description includes the name of the data set, the number of features and the total number of examples. Nineteen of these data sets are from the UCI repository (Blake & Merz, 1998) while the remaining eleven, denoted in Table 2.3 by a "+", were contributed from researchers at AT&T (Cohen, 1995; Cohen & Singer, 1999). Except for the soybean-large data set, all of the data sets have exactly two classes (we have a two-class version of the splice-junction data set).

#	Data Set	Features	Size	#	Data Set	Features	Size
1	adult	14	21,280	16	market1+	10	3,180
2	bands	39	538	17	market2+	10	11,000
3	blackjack+	4	15,000	18	move+	10	3,028
4	breast-wisc	9	699	19	network1+	30	3,577
5	bridges	7	101	20	network2+	35	3,826
6	coding	15	20,000	21	ocr+	576	2,688
7	crx	15	690	22	promoters	57	106
8	german	20	1,000	23	sonar	60	208
9	heart-hungarian	13	293	24	soybean-large	35	682
10	hepatitis	19	155	25	splice-junction	60	3,175
11	horse-colic	23	300	26	ticket1+	78	556
12	hypothyroid	25	3,771	27	ticket2+	53	556
13	kr-vs-kp	36	3,196	28	ticket3+	61	556
14	labor	16	57	29	vote	16	435
15	liver	6	345	30	weather+	35	5,597

Table 2.3: Data Sets used to Study Small Disjuncts

The experiments involving class distribution alter the class distribution of the training data. The methodology for altering the class distribution, described in detail in Chapter 6, involves discarding training examples so that the desired class distribution is achieved. To ensure that there are sufficient training examples for learning even after these examples are discarded, extremely small data sets are excluded from the study of class distribution.

Table 2.4 provides a brief description of the twenty-six data sets used to study class distribution. Twenty of these data sets are from the UCI repository and five data sets, again identified with a "+", were contributed by researchers from AT&T. The *phone* data set was supplied by the author. The descriptions in Table 2.4 also specify the class distribution for each data set, in terms of the percentage of minority-class examples contained within the data set. The data sets are listed in the table in order of decreasing class imbalance, a convention used throughout this thesis for experiments concerning class distribution.

#	Data Set	% Minority	Features	Size	#	Data Set	% Minority	Features	Size
1	letter-a*	3.9	17	20,000	14	network2	27.9	35	3,826
2	pendigits*	8.3	16	13,821	15	yeast*	28.9	8	1,484
3	abalone*	8.7	8	4,177	16	network1+	29.2	30	3,577
4	sick-euthyroid	9.3	25	3,163	17	car*	30.0	6	1,728
5	connect-4*	9.5	42	11,258	18	german	30.0	20	1,000
6	optdigits*	9.9	64	5,620	19	breast-wisc	34.5	9	699
7	covertype*	14.8	54	581,102	20	blackjack+	35.6	4	15,000
8	solar-flare*	15.7	10	1,389	21	weather+	40.1	35	5,597
9	phone	18.2	13	652,557	22	bands	42.2	39	538
10	letter-vowel*	19.4	17	20,000	23	market1+	43.0	10	3,181
11	contraceptive*	22.6	9	1,473	24	crx	44.5	15	690
12	adult	23.9	14	48,842	25	kr-vs-kp	47.8	36	3,196
13	splice-junction*	24.1	60	3,175	26	move+	49.9	10	3,029

Table 2.4: Data Sets used to Study Class Distribution

In order to simplify the presentation and the analysis of results, for the class distribution experiments data sets with more than two classes are mapped to two-class problems. This is accomplished by designating one of the original classes, typically the least frequently occurring class, as the minority class and then mapping the remaining classes into the majority class. The data sets that originally contained more than two classes are identified with an asterisk (*) in Table 2.4. The *letter-a* and *letter-vowel* data sets were created from the *letter-recognition* data set by assigning those examples labeled with either the letter “a”, or with a vowel, to the minority class.

2.4 Summary

This chapter summarizes the basic background material needed to understand the experiments described in this thesis. It describes the two classifier induction programs used in this thesis, C4.5 and Ripper, the metrics used to evaluate the performance of the classifiers induced using these programs, and the data sets used to study the effect of small disjuncts and class distribution on learning. Because of deficiencies with accuracy, classifier performance is measured using both accuracy and the area under the ROC curve.

Chapter 3

The Problem with Small Disjuncts

"The best way to escape from a problem is to solve it."
- Alan Saporta

"Challenges are what make life interesting;
overcoming them is what makes life meaningful."
- Joshua J. Marine

Small disjuncts and their impact on learning are investigated in this and the following chapter. Results from previous research studies indicate that classification errors tend to be concentrated heavily in the small disjuncts (Holte et al., 1989; Ali & Pazzani, 1992; Danyluk & Provost, 1993; Ting, 1994; Weiss, 1995; Weiss & Hirsh, 1998; Carvalho & Freitas, 2000). These studies, however, only analyze one or two data sets in detail. In this chapter the problem with small disjuncts is measured much more comprehensively, by measuring the degree to which errors are concentrated toward the small disjuncts for classifiers induced from thirty data sets, using both C4.5 and Ripper. The results show that most classifiers exhibit the problem with small disjuncts, albeit to different degrees.

Chapter 4 then analyzes how factors such as pruning, training-set size, noise and class imbalance each affect small disjuncts and the distribution of errors across disjuncts. This analysis will also provide a better understanding of how each of these factors affects learning. Finally, Chapter 5 discusses related research.

3.1 The Formation of Small Disjuncts in Classifier Learning

We begin this chapter by providing important background concerning the formation of small disjuncts. In particular, we discuss two reasons why small disjuncts are formed and suggest reasons for why they may be more error prone than large disjuncts. This section therefore is useful in providing a context in which to interpret the results from the empirical study of small disjuncts, presented in this chapter and Chapter 4.

3.1.1 An Inability to Express the Target Concept

Small disjuncts may be formed because the classifier cannot readily express the target concept. In this thesis the study of small disjuncts relies primarily on C4.5, a decision tree learner, so we begin by examining the expressive power of such a learner. Decision tree learners test (i.e., branch on) one attribute at a time and hence form decision boundaries by making axis-parallel cuts in the instance space. Thus, decision tree learners can only approximate concepts that are not expressible using only axis-parallel cuts. This situation is depicted graphically in Figure 3.1.

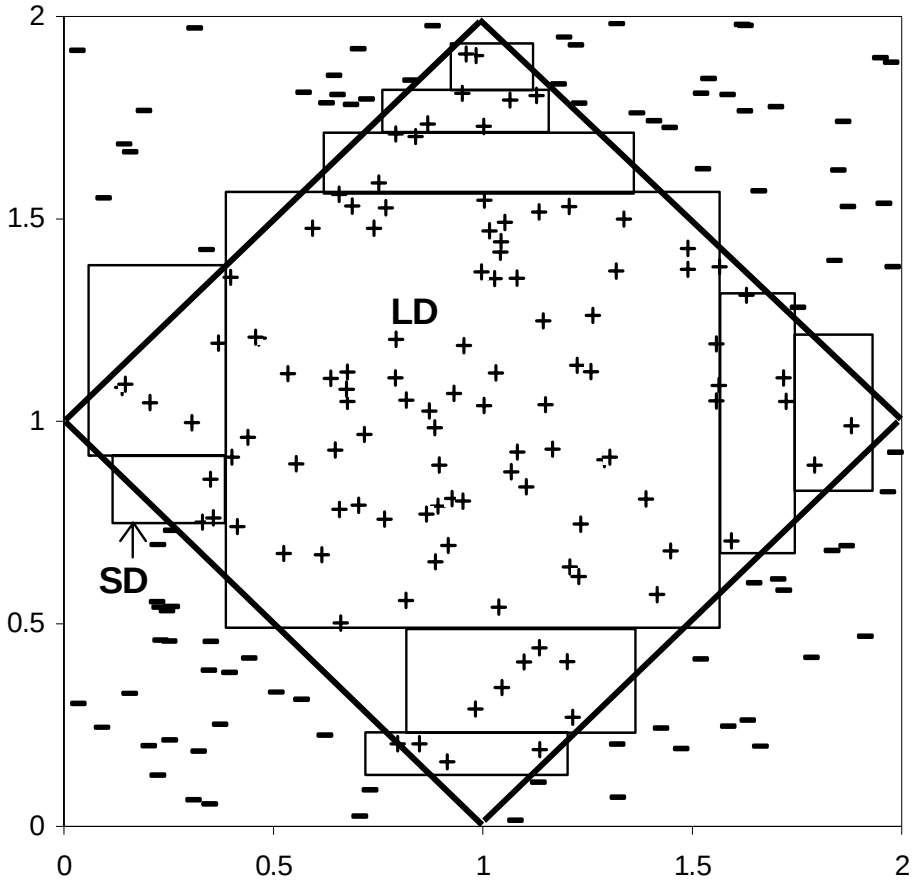

Figure 3.1: A Concept that cannot be Perfectly Learned by a Decision Tree Learner

The target concept in Figure 3.1 is a square that is centered about the point (1, 1) and rotated forty-five degree with respect to the x-axis. The region covered by each disjunct in the induced classifier is represented by a rectangle. The largest of these rectangles, is labeled LD, indicating it is the largest disjunct. The points labeled "+" and "-" represent examples belonging to the positive and negative classes. A total of 200 examples were randomly selected (using a random-number generator) over the entire instance space.

Figure 3.1 makes it clear that no rectangle parallel to the x-axis can cover the target concept. The largest disjunct covers most, but not all, of the target concept. In a very real sense the remaining disjuncts are formed in an attempt to learn the non-axis parallel

boundaries of the target concept. As can be seen, these disjuncts tend to be smaller disjuncts. It would take an infinite number of rectangles to perfectly learn the true boundaries of the target concept.

The disjuncts shown in Figure 3.1 correctly classify all of the training examples. Most of the decision boundaries associated with the disjuncts have some "play"—they could be moved in or out without misclassifying any training examples. These boundaries were generally set to fall about halfway between the closest positive and negative classes. Note that in this example the learned concept does not cover the entire target concept. Given the assumption that the examples are uniformly distributed over the space, the error rate for each disjunct is the percentage of its area that falls outside of the target concept. Based on this, it is clear that largest disjunct has a very low error rate, while the small disjuncts have a high error rate. In general, those disjuncts that are centered near the decision boundary of the target concept will tend to have a higher error rate. This provides one explanation for why small disjuncts may have a higher error rate than large disjuncts.

3.1.2 A Target Concept With Small Subconcepts

A target concept with small subconcepts may also lead to the formation of small disjuncts. To see this, consider the target concept in Figure 3.2, which has nine subconcepts. The examples are again assumed to be uniformly distributed over the space and the 1,000 examples that are displayed in the figure were randomly generated (the positive examples are represented by a shaded box to make them more noticeable). Eight of the subclasses are defined by squares where each side is 0.2 units long (1/10 the length of the entire space). Consequently, each of these eight small subconcepts is expected to cover 1% of

the examples. The large subconcept is defined by a square with length 0.6 and hence is expected to cover 9% of all examples. These squares are all axis-parallel, so the inability of a decision tree learner to form decision boundaries that are not axis-parallel is not an issue.

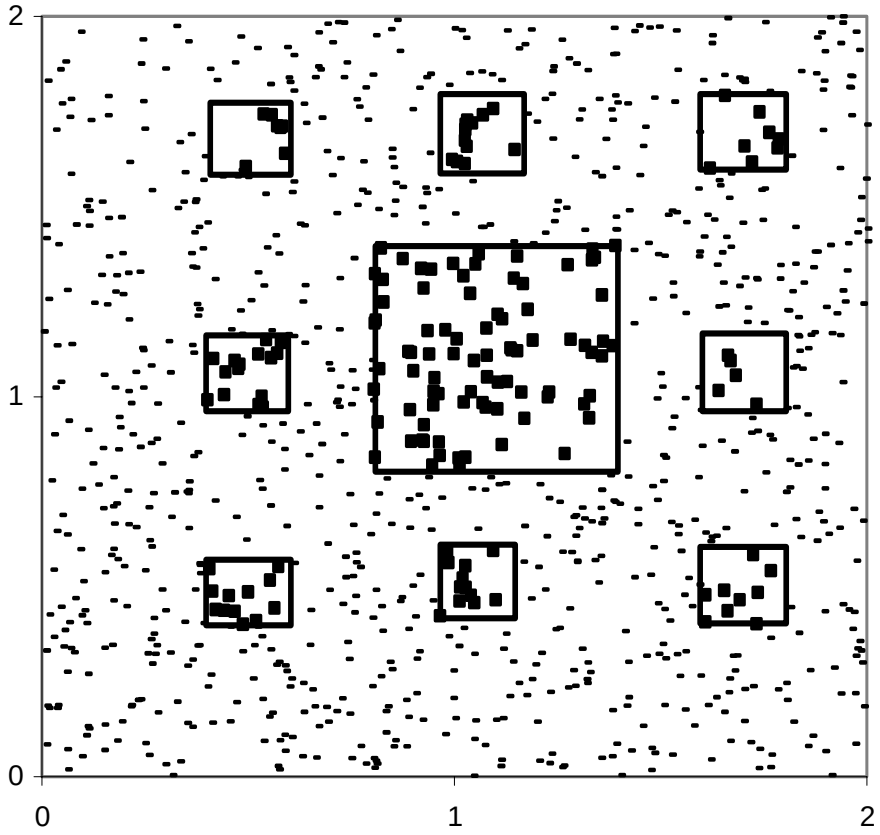

Figure 3.2: A Target Concept with Many Small Subclasses

The disjuncts that might be formed to cover the target concept are not shown in Figure 3.2, but it should be clear that each subconcept could be learned using a single disjunct. In this case there would be one large disjunct and eight small disjuncts, where the small disjuncts would tend to have similar sizes (the differences would be due to the exact location of the learned decision boundaries). Thus, small subconcepts represent a second rea-

son that small disjuncts are formed. Note that this scenario applies to any learner that forms disjunctive concepts—not just decision tree learners.

The small disjuncts that are formed to cover the eight small subconcepts in Figure 3.2 will tend to have a higher error rate than the disjunct formed to cover the large subconcept. To see this, first assume that for any of the subconcepts, the learned boundary is Δ away from the true boundary. For simplicity, we assume that the value of Δ is the same for all four sides of the learned decision boundary. The value of Δ determines how well the subconcept is learned. If we assume that Δ is, on average, the same for the large disjunct (learned to cover the large subconcept) as for the small disjuncts (learned to cover the small subconcepts), then the large disjunct will have a much lower error rate—because it covers a much larger region that is correctly learned.

The only way the error rate for the small disjuncts would not be greater than the error rate for the large disjunct is if Δ were smaller for the small disjuncts. However, there is no reason to believe that this would be so. In fact, there is some reason to believe that Δ would be smaller for the large disjunct. To see this, consider the task of learning the bottom-most decision boundary (or any boundary) for each of the subconcepts. Given that there is no noise and the subconcepts can be learned perfectly, the disjunct should cover no minority-class training examples. Thus, the bottom-most boundary is determined by the negative and positive example that is nearest, vertically, to the target subconcept's bottom boundary. Since the boundary is three times longer for the large subconcept, given the assumption that examples are uniformly distributed over the space, we expect the closest positive and negative examples to generally be closer to the true boundary for the large subconcept.

3.2 An Example of the Problem with Small Disjuncts

We next turn to the problem with small disjuncts. The problem with small disjuncts is that they generally have a much higher error rate than large disjuncts and contribute a substantial percentage of all classification errors. In order to demonstrate the problem with small disjuncts, a simple example is presented in Figure 3.3. This figure shows the performance of classifiers induced by C4.5 from the Vote data set. This figure shows how the correctly and incorrectly classified examples are distributed across the disjuncts in the induced classifier.

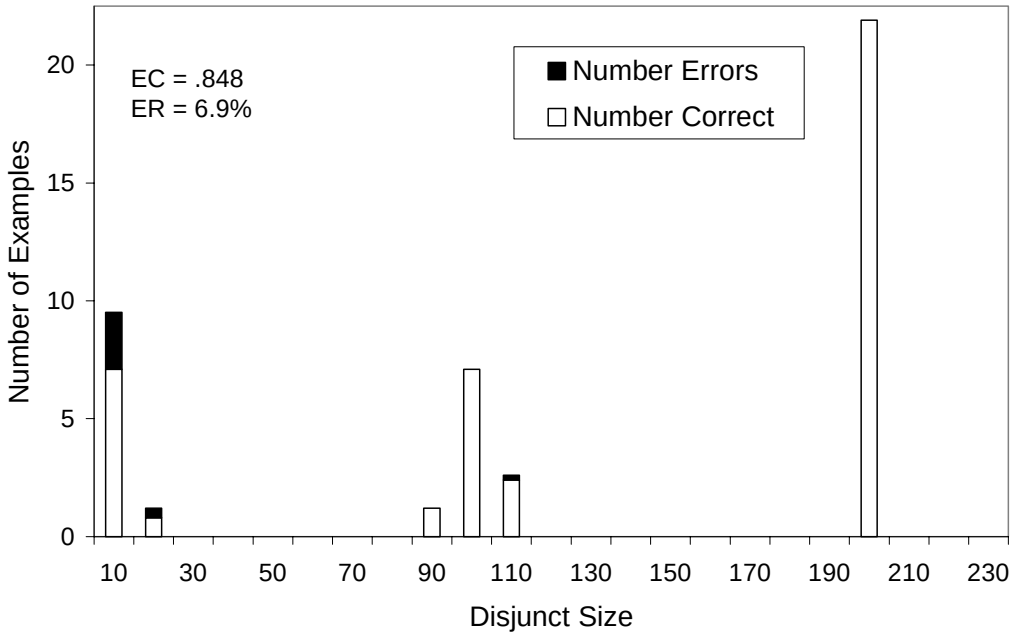

Figure 3.3: Distribution of Examples for Vote Data Set

The x-axis in Figure 3.3 specifies the size of a disjunct while the y-axis specifies the number of correctly and incorrectly classified test examples covered by disjuncts with that size. Because there are typically no disjuncts associated with each specific disjunct size, the x-values are placed into bins, in order to make the results more comprehensible. Each bin in the histogram in Figure 3.3 covers ten sizes of disjuncts. Thus, the leftmost

bin shows that those disjuncts that correctly classify 0-9 training examples cover 9.5 test examples, of which 7.1 are classified correctly and 2.4 classified incorrectly (fractional values occur because the results are averaged over 10 cross-validated runs).

Figure 3.3 shows that the test examples from the vote data set are covered by disjuncts with a variety of sizes. The results in Figure 3.3 indicate that the smaller disjuncts have a much higher error rate than the larger disjuncts and that the small disjuncts contribute most of the total test errors. An examination of the detailed data underlying Figure 3.3 shows that the errors are skewed even more toward the smallest disjuncts than suggested by the figure—75% of the errors in the leftmost bin come from disjuncts of size 0 and 1 (disjuncts of size 0 occur because when C4.5 splits a node N using a feature f, the split will branch on all *possible* values of f*)*. One may also be interested in the distribution of disjuncts by disjunct size. The classifier associated with Figure 3.1 is made up of fifty disjuncts, of which forty-five are associated with the leftmost bin (i.e. have a disjunct size less than 10). The error rate of the classifier is 6.9%. The error concentration (EC) for this classifier is .848 (this metric is described in the next section).

The vote data set is used throughout Chapter 4 to show how pruning, training-set size and noise affect small disjuncts. Thus, the distribution of errors in Figure 3.3 serves as a baseline for future comparison.

3.3 Error Concentration

The study of small disjuncts requires the ability to measure and represent classification performance with respect to disjunct size. The graphical representation in Figure 3.3 is commonly used in research on small disjuncts. However, there are two problems with the way information is represented in the figure. First, the degree to which errors are

concentrated toward the small disjuncts is not clear. That is, while Figure 3.3 shows that the errors are heavily concentrated in the smaller disjuncts, how do we quantify this concentration of errors? Secondly, how do we compare the distribution of errors between different classifiers?

The inadequacies with the representation in Figure 3.3 can be addressed by plotting the percentage of total test errors versus the percentage of correctly classified test examples contributed by a set of disjuncts. The curve in Figure 3.4 is generated by starting with the smallest disjunct from the classifier and progressively adding larger disjuncts. This curve shows, for example, that disjuncts with size 0-4 cover 5.1% of the correctly classified test examples but 73% of the total test errors. The line Y=X represents a classifier in which classification errors are distributed throughout the disjuncts without regard to the size of the disjunct. Since the "error concentration" curve in Figure 3.4 falls above the line Y=X, the errors produced by this classifier are more concentrated toward the smaller disjuncts than to the larger disjuncts.

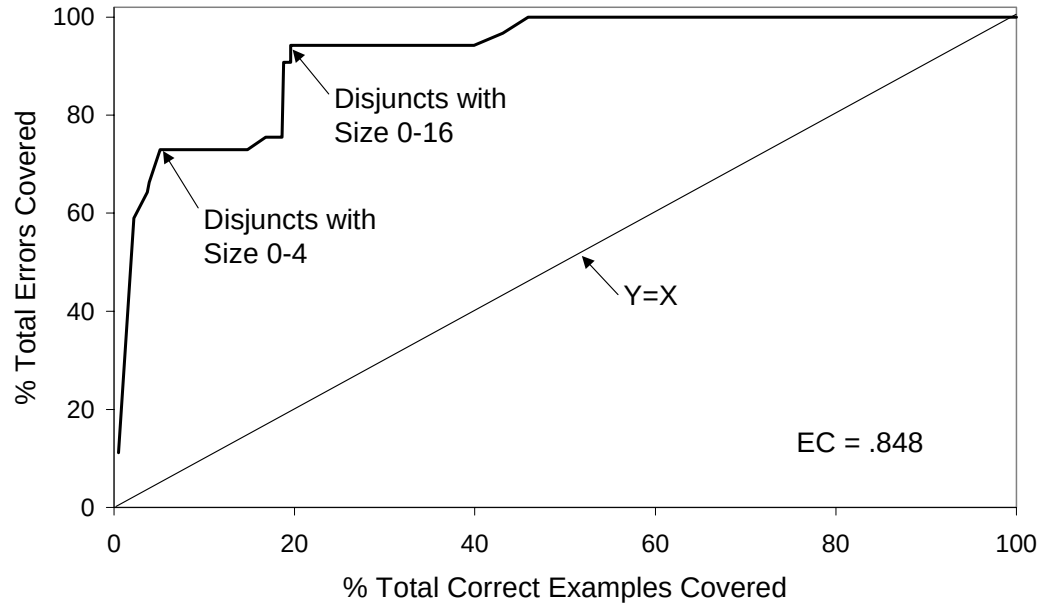

Figure 3.4: An Error Concentration Curve

To make it easy to compare the degree to which errors are concentrated toward the smaller disjuncts for different classifiers, we introduce *error concentration.* The error concentration (EC) of a classifier is defined as the fraction of the area above the line Y=X that falls below its error concentration curve. Using this scheme, the higher the error concentration, the more concentrated the errors are toward the smaller disjuncts. Error concentration may range from a value of +1, which indicates that the smallest disjunct(s) covers all test errors, before even a single correctly classified test example is covered, to a value of –1, which indicates that the largest disjunct(s) covers all test errors, after all correctly classified test examples have been covered. The error concentration for the classifier described in Figure 3.4 is .848, which indicates that the errors are highly concentrated toward the small disjuncts.

Based on previous research, which indicates that small disjuncts have higher error rates than large disjuncts, one would expect the error concentration of most classifiers to be greater than 0. This is investigated in detail later in this chapter, by measuring the error concentration of classifiers induced from thirty data sets.

3.4 Experimental Methodology

Some of the experimental methodology was described in Chapter 2, which included a description of the classifier induction programs, the data sets to be studied and the metrics for evaluating classifier performance. In this section we briefly describe some of the methodology that is specific to the study of small disjuncts.

All experiments related to small disjuncts utilize C4.5, while many, but not all experiments are repeated using Ripper to ensure the generality of the results. All experiments for studying small disjuncts employ ten-fold cross validation. The associated ex-

perimental results, presented later in this chapter and in Chapter 4, are therefore based on averages over ten runs.

Pruning tends to eliminate most small disjuncts and, for this reason, research on small disjuncts generally disables pruning (Holte, et al., 1989; Danyluk & Provost, 1993; Weiss, 1995; Weiss & Hirsh, 1998). If this were not done, then pruning would mask the problem with small disjuncts. While this means that the analyzed classifiers are not the same as the ones that would be generated using the learners in their standard configurations, these results are nonetheless important, since the performance of an unpruned classifier constrains the performance of a pruned classifiers. However, in this thesis both unpruned and pruned classifiers are analyzed, for both C4.5 and Ripper. This makes it possible to analyze the effect that pruning has on small disjuncts and to evaluate pruning as a strategy for addressing the problem with small disjuncts. As the results for pruning in Chapter 4 will show, the problem with small disjuncts is still evident after pruning, although to a lesser extent.

Except where specifically noted, all results are based on the use of C4.5 and Ripper with their pruning strategies disabled. For C4.5, when pruning is disabled the –m 1 option is also used, to ensure that C4.5 does not stop splitting a node before the node contains examples belonging to a single class (the default is –m 2). Ripper is configured to produce unordered rules so that it does not produce a single default rule to cover the majority class.

3.5 Measuring the Impact of Small Disjuncts: Results and Analysis

This section investigates the impact that small disjuncts have on learning. This is accomplished by describing and analyzing the performance of the classifiers induced by C4.5

and Ripper from the thirty data sets listed in Table 2.3, using error concentration to measure the distribution of errors by disjunct size in the induced classifiers. This study of the distribution of errors in the induced classifiers is much more thorough than previous studies (Holte et al., 1989; Ali & Pazzani, 1992; Danyluk & Provost, 1993; Weiss, 1995; Weiss & Hirsh, 1998; Carvalho & Freitas, 2000), which analyzed only one or two data sets and had no measure equivalent to error concentration that could be used to compare the distribution of error across classifiers.

3.5.1 Experimental Results

The performance of the classifiers induced by C4.5 and Ripper are described in Table 3.1 and Table 3.2, respectively. The results are listed in order of decreasing error concentration, so that the data sets near the top of the tables have the errors most heavily concentrated toward the small disjuncts.

Tables 3.1 and 3.2 provide several pieces of information in addition to the error concentration of the induced classifier. The tables first list the EC rank of the classifier and the name of the data set. This is followed by the size of the data set and the size of the largest disjunct in the induced classifier. The next three columns then provide some very specific information about how the errors are distributed based on disjunct size. This includes the percentage of the total test errors that are contributed by the smallest disjuncts that collectively cover 10% (20%) of the correctly classified test examples, followed by the percentage of all correctly classified examples that are covered by the smallest disjuncts that collectively cover half of the total errors. These last three values are reported because error concentration is a *summary* statistic, and as such, may be somewhat abstract and difficult to interpret. The next two columns describe the overall performance

of the classifier, first using error rate and then AUC, the area under the ROC curve (the AUC value is not computed for the soybean-large data set because it contains more than two classes). Finally, the last column specifies the error concentration. Recall that larger AUC values indicate improved classifier performance.

EC Rank	Data Set Name	Data Set Size	Largest Disjunct	% Errors at 10% Correct	% Errors at 20% Correct	% Correct at 50% Errors	Error Rate	AUC	Error Conc.
1	kr-vs-kp	3,196	669	75.0	87.5	1.1	0.3	.987	.874
2	hypothyroid	3,771	2,697	85.2	90.7	0.8	0.5	.934	.852
3	vote	435	197	73.0	94.2	1.9	6.9	.947	.848
4	splice-junction	3,175	287	76.5	90.6	4.0	5.8	.885	.818
5	ticket2	556	319	76.1	83.0	2.7	5.8	.816	.758
6	ticket1	556	366	54.8	90.5	4.4	2.2	.964	.752
7	ticket3	556	339	60.5	84.5	4.6	3.6	.870	.744
8	soybean-large	682	56	53.8	90.6	9.3	9.1	N/A	.742
9	breast-wisc	699	332	47.3	63.5	10.7	5.0	.945	.662
10	ocr	2,688	1,186	52.1	65.4	8.9	2.2	.825	.558
11	hepatitis	155	49	30.1	58.0	17.2	22.1	.717	.508
12	horse-colic	300	75	31.5	52.1	18.2	16.3	.776	.504
13	crx	690	58	32.4	61.7	14.3	19.0	.818	.502
14	bridges	101	33	15.0	37.2	23.2	15.8	.693	.452
15	heart-hungarian	293	69	31.7	45.9	21.9	24.5	.793	.450
16	market1	3,180	181	29.7	48.4	21.1	23.6	.783	.440
17	adult	21,280	1,441	28.7	47.2	21.8	16.3	.805	.424
18	weather	5,597	151	25.6	47.1	22.4	33.2	.715	.416
19	network2	3,826	618	31.2	46.9	24.2	23.9	.702	.384
20	promoters	106	20	32.8	48.7	20.6	24.3	.608	.376
21	network1	3,577	528	26.1	44.2	24.1	24.1	.697	.358
22	german	1,000	56	17.8	37.5	29.4	31.7	.593	.356
23	coding	20,000	195	22.5	36.4	30.9	25.5	.632	.294
24	move	3,028	35	17.0	33.7	30.8	23.5	.658	.284
25	sonar	208	50	15.9	30.1	32.9	28.4	.649	.226
26	bands	538	50	65.2	65.2	54.1	29.0	.592	.178
27	liver	345	44	13.7	27.2	40.3	34.5	.562	.120
28	blackjack	15,000	1,989	18.6	31.7	39.3	27.8	.683	.108
29	labor	57	19	33.7	39.6	49.1	20.7	.673	.102
30	market2	11,000	264	10.3	21.6	45.5	46.3	.540	.040

Table 3.1: Error Concentration Results for C4.5

EC Rank	C4.5 Rank	Data Set Name	Data Set Size	Largest Disjunct	% Errors at 10% Correct	% Errors at 20% Correct	% Correct at 50% Errors	Error Rate	AUC	Error Conc.
1	2	hypothyroid	3,771	2,696	96.0	96.0	0.1	1.2	.980	.898
2	1	kr-vs-kp	3,196	669	92.9	92.9	2.2	0.8	.999	.840
3	6	ticket1	556	367	69.4	95.2	1.6	3.5	.985	.802
4	7	ticket3	556	333	61.4	81.5	5.6	4.5	.976	.790
5	5	ticket2	556	261	71.0	91.0	3.2	6.8	.920	.782
6	3	vote	435	197	75.8	75.8	3.0	6.0	.983	.756
7	4	splice-junction	3,175	422	62.3	76.1	7.9	6.1	.975	.678
8	9	breast-wisc	699	355	68.0	68.0	3.6	5.3	.973	.660
9	8	soybean-large	682	61	69.3	69.3	4.8	11.3	N/A	.638
10	10	ocr	2,688	804	50.5	62.2	10.0	2.6	.979	.560
11	17	adult	21,280	1,488	36.9	56.5	15.0	19.7	.812	.516
12	16	market1	3,180	243	32.2	57.8	16.9	25.0	.839	.470
13	12	horse-colic	300	73	20.7	47.2	23.9	22.0	.858	.444
14	13	crx	690	120	32.5	50.3	19.7	17.0	.914	.424
15	15	hungarian-heart	293	67	25.8	44.9	24.8	23.9	.828	.390
16	26	bands	538	62	25.6	36.9	29.2	21.9	.845	.380
17	25	sonar	208	47	32.6	41.2	23.9	31.0	.793	.376
18	23	coding	20,000	206	22.6	37.6	29.2	28.2	.794	.374
19	18	weather	5,597	201	23.8	42.1	24.8	30.2	.781	.356
20	24	move	3,028	45	25.9	44.5	25.6	32.1	.708	.342
21	14	bridges	101	39	41.7	41.7	35.5	14.5	.755	.334
22	20	promoters	106	24	20.0	50.6	20.0	19.8	.871	.326
23	11	hepatitis	155	60	19.3	47.7	20.8	20.3	.725	.302
24	22	german	1,000	99	12.1	31.2	35.0	30.8	.711	.300
25	19	network2	3,826	77	25.6	45.9	22.9	23.1	.646	.242
26	27	liver	345	28	28.2	37.4	32.0	34.0	.708	.198
27	28	blackjack	15,000	1,427	12.3	24.2	42.3	30.2	.695	.108
28	21	network1	3,577	79	18.9	29.7	46.0	23.4	.657	.090
29	29	labor	57	21	0.0	55.6	18.3	24.5	.620	-.006
30	30	market2	11,000	55	10.4	21.1	49.8	48.8	.516	-.018

Table 3.2: Error Concentration Results for Ripper

As an example of how to interpret the results in these tables, consider the entry for the *kr-vs-kp* data set in Table 3.1. The error concentration for the classifier induced from this data set is .874. Furthermore, the smallest disjuncts that collectively cover 10% of the correctly classified test examples contribute 75% of the total test errors, while the smallest disjuncts that contribute half of the total errors cover only 1.1% of the total correctly classified examples. These measurements indicate that the errors are very highly concentrated toward the smaller disjuncts.

The results for C4.5 and Ripper show that although the error concentration values are, as expected, almost always positive, the values vary widely, indicating that the induced classifiers suffer from the problem of small disjuncts to varying degrees. In particular, note that for a few data sets, such as the *labor* and *market2* data sets, the error concentration is near zero—and for Ripper the error concentration for these data sets is actually negative. This shows something new—that the problem with small disjuncts is not observed for all classifiers.

3.5.2 Comparison of Results for Classifiers Induced using C4.5 and Ripper

The classifiers induced using Ripper have a slightly smaller average error concentration than those induced using C4.5 (.445 vs. .471), indicating that the classifiers induced by Ripper have the errors spread slightly more uniformly across the disjuncts. Overall, Ripper and C4.5 tend to generate classifiers with similar error concentration values. This can be seen by comparing the EC rank in Table 3.2 for Ripper (column 1) with the EC rank for C4.5 (column 2).

This relationship between C4.5's and Ripper's error concentrations can be seen even more clearly using the scatter plot in Figure 3.5, where each point represents the error concentration for a single data set. Since the points in Figure 3.5 are clustered around the line Y=X, both learners tend to produce classifiers with similar error concentrations and hence tend to suffer from the problem with small disjuncts to similar degrees. The agreement is especially close for the most interesting cases, where the error concentrations are large—the largest ten error concentration values in Figure 3.5, for both C4.5 and Ripper, are generated by the same ten data sets. The Spearman rank correlation (Kendall & Gibbons, 1990) can be used to summarize the correlation between the error concentra-

tions of the two learners. In this case the rank correlation is .87, indicating that the values are highly correlated (the closer to 1 the greater the correlation).

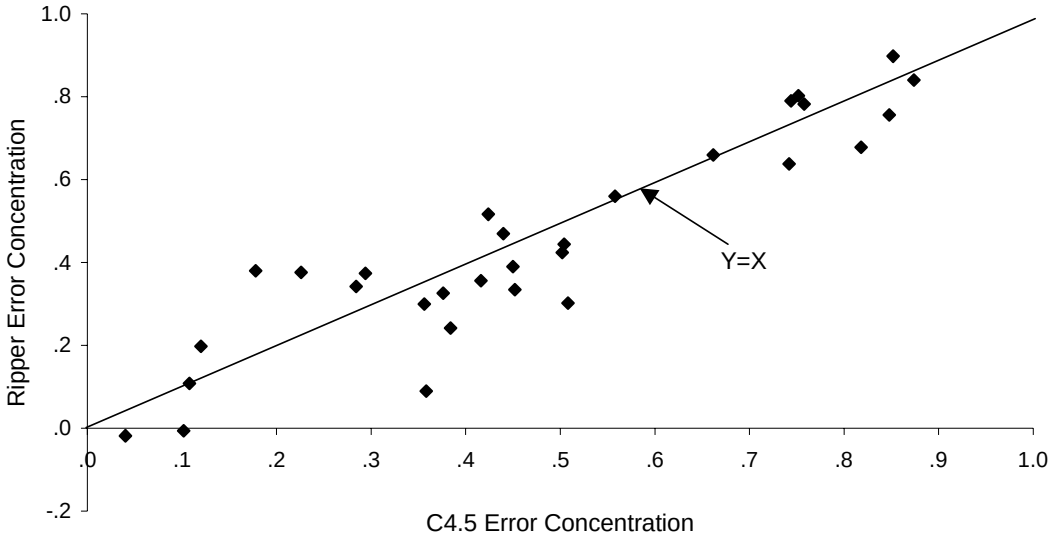

Figure 3.5: Comparison of C4.5 and Ripper Error Concentrations

We can also compare the performance of the classifiers induced by C4.5 and Ripper. With respect to classification accuracy, the two learners perform similarly, although C4.5 performs slightly better. In particular, C4.5 outperforms Ripper on 18 of the 30 data sets, with an average error rate of 18.4% vs. 19.0% (the results in the next chapter will show that Ripper slightly outperforms C4.5 when pruning is used). However, when AUC is used to compare classifier performance, the results indicate that Ripper consistently outperforms C4.5. In this case Ripper outperforms C4.5 in 25 of 29 cases, with an average AUC of .820 vs. .754 for C4.5.

3.5.3 Analysis of Results

The results in Table 3.1 and Table 3.2 indicate that, for both C4.5 and Ripper, there is a relationship between the error concentration (EC) of the induced classifier and its error

rate (ER) and AUC. For example, these results show that, for the thirty data sets, when the induced classifier has an error rate less than 12%, then the error concentration is always greater than .50. More generally, error concentration is positively correlated with classifier performance. This can be seen by computing the Spearman rank correlation between error concentration and classifier performance. The Spearman rank correlation between error concentration and accuracy (i.e., 1 – error rate) is .86 for C4.5 and .80 for Ripper, while the Spearman rank correlation between error concentration and AUC is .91 for C4.5 and .94 for Ripper. These results indicate "good" correlation in all cases. AUC may correlate better with EC than accuracy due to the fact that AUC, as described in Section 2.2.3, is not affected by the underlying class distribution (Perlich, Provost & Siminoff).

Based on error concentration and classifier performance, the induced classifiers can be placed into the following three categories (the boundaries between each category are somewhat arbitrary):

1. High-EC includes data sets 1-10 for C4.5 and Ripper
2. Medium-EC includes data sets 11-22 for C4.5 and 11-24 for Ripper
3. Low-EC includes data sets 23-30 for C4.5 and 25-30 for Ripper

The classifiers in the high-EC category generally outperform those in the medium-EC category, which in turn generally outperform those in the low-EC category. It is interesting to note that for those data sets in the high-EC category, the largest disjunct generally covers a very large portion of the total training examples. As an example, consider the hypothyroid data set. Of the 3,394 examples (90% of the total data) used for training, nearly 2,700 of these examples, or 79%, are covered by the largest disjunct induced by

C4.5 and Ripper. To see that these large disjuncts are extremely accurate, consider the vote data set, which falls within the same category. The data used to generate the distribution of errors for the vote data set, shown previously in Figure 3.3, indicate that the largest disjunct, which covers 23% of the total training examples, does not contribute a single error when used to classify the test data. These observations lead us to speculate that target concepts that can be learned well (i.e., the induced classifier has a low error rate) are often made up of "large" subconcepts that cover many examples that lead to highly accurate large disjuncts—and therefore to classifiers with very high error concentrations. Target concepts that are difficult to learn, on the other hand, either are not made up of large subconcepts, or, due to limitations with the expressive power of the learner, these large subconcepts cannot be represented using large disjuncts. This leads to classifiers without very large, highly accurate, disjuncts and with many small disjuncts. These classifiers tend to have much smaller error concentrations and high error rates.

3.6 Summary

This chapter began with a discussion of why small disjuncts are formed and why they might be expected to be more error prone than large disjuncts. This was followed by a simple example that demonstrates that small disjuncts are much more error prone than large disjuncts and are responsible for many of the test errors. A new metric, error concentration, was then introduced, which summarizes the degree to which errors are concentrated toward the small disjuncts. After some methodological issues were addressed, error concentration was measured for classifiers induced by C4.5 and Ripper. The results indicate that errors are highly concentrated in the small disjuncts in many cases—but that this is not true in all cases.

Analysis of the results showed some obvious patterns. In particular, the results indicate that classifiers with low error rates and large AUC values tend to have high error concentrations while classifiers with high error rates and low AUC values tend to have low error concentrations. These results and the underlying data indicate that well-learned concepts generally include highly accurate, very large disjuncts. We conclude that in many cases those target concepts that can be well learned include subconcepts that cover many examples.

Chapter 4

Factors Affecting Small Disjuncts and Error Concentration

The pure and simple truth is rarely pure and never simple.

- Oscar Wilde

This chapter analyzes how factors such as pruning, training-set size, noise, and class imbalance affect the formation of small disjuncts and the distribution of errors in the learned classifiers. In addition to shedding light into how small disjuncts affect learning, this analysis will also yield a better understanding of these important factors and the mechanism by which they influence learning. Some of the more striking results in this chapter involve pruning. Pruning is shown to disproportionately eliminate small disjuncts. Because this causes the errors to be distributed more uniformly throughout the disjuncts, pruning is shown to hurt the accuracy of large disjuncts. However, pruning is shown to be quite effective at combating the effects of noise.

4.1 Pruning

Consistent with prior research on small disjuncts, the experimental results in Chapter 3 were generated using C4.5 and Ripper with their pruning strategies disabled. Pruning is not used when studying small disjuncts because of the belief that it disproportionately eliminates small disjuncts from the induced classifier and thereby obscures the very phe-

nomenon to be studied. However, because pruning is employed by many classifier induction programs, it is worthwhile to understand how it affects small disjuncts and the distribution of errors across disjuncts—as well as how effective it is at addressing the problem with small disjuncts.

4.1.1 A Simple Example

The analysis of pruning begins with a simple, illustrative, example. The distribution of errors for the classifiers induced by C4.5 from the vote data set are shown in Figures 4.1 and 4.2, without pruning and with pruning, respectively. A comparison of these two figures shows how pruning affects the distribution of errors with respect to disjunct size.

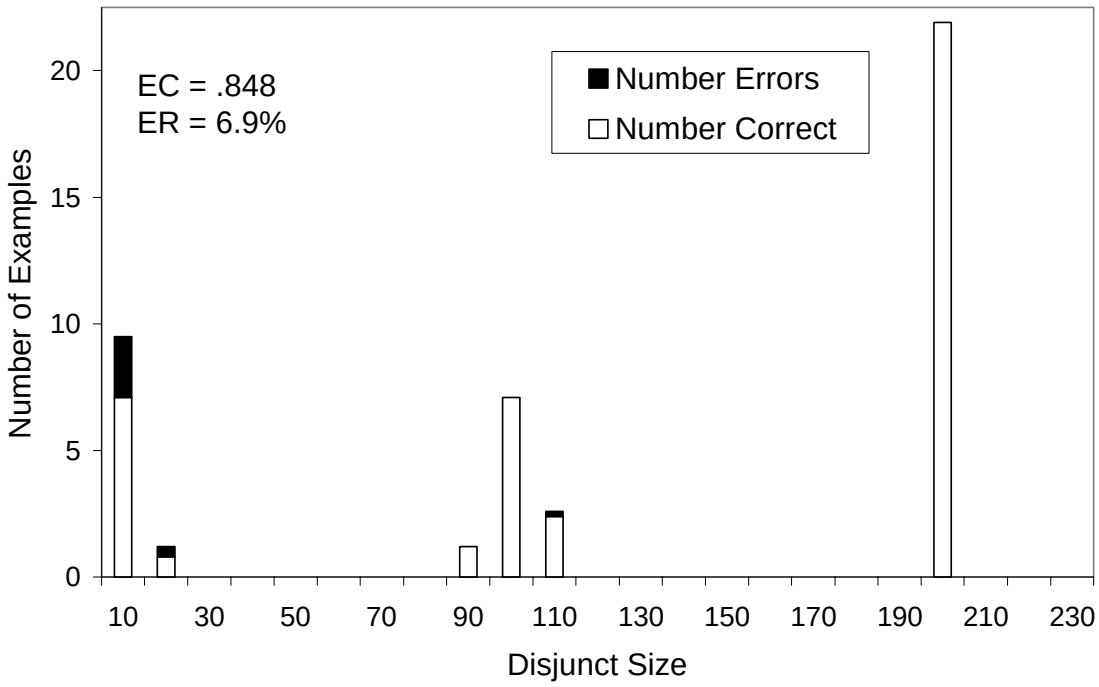

Figure 4.1: Distribution of Errors without Pruning

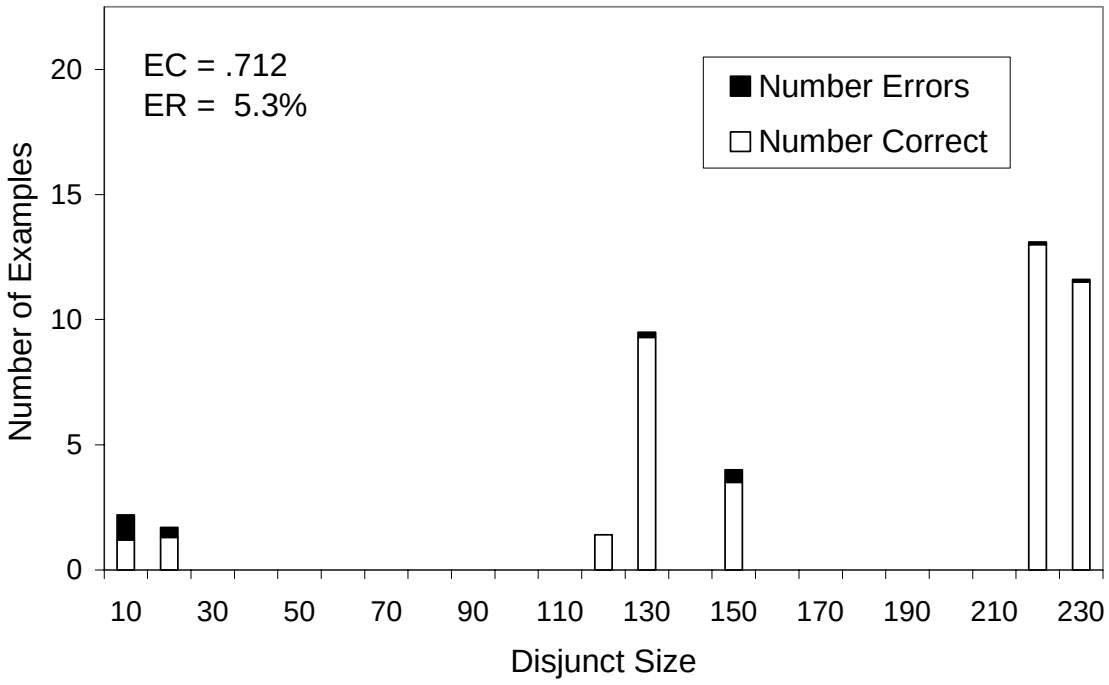

Figure 4.2: Distribution of Errors with Pruning

A visual comparison of the figures shows that with pruning the errors are less concentrated toward the small disjuncts—which is confirmed by the reduction in error concentration from .848 to .712. By comparing the two left-most bins in each figure it is also apparent that with pruning far fewer examples are classified by disjuncts with size 0-9 and 10-19. The reason for this is that the distribution of disjuncts has changed. The underlying data indicate that without pruning the induced classifiers typically (i.e., over the 10 cross-validated runs) contain 48 disjuncts, of which 45 are of size 10 or less, while with pruning only 10 disjuncts remain, of which 7 have size 10 or less. So, in this case pruning eliminates 38 of the 45 disjuncts with size 10 or less. This confirms the assumption that pruning eliminates many, if not most, small disjuncts. Most of the emancipated examples—those examples that would have been classified by the eliminated disjuncts—are now classified by larger disjuncts. Because a comparison of Figures 4.1 and 4.2

shows that, with pruning, the large disjuncts have more errors, we conclude that by eliminating many small disjuncts, pruning degrades the classification performance of the highly accurate large disjuncts.

It is important to note that even with pruning the error concentration of the induced classifiers is still quite positive (.712), indicating that errors are still heavily concentrated toward the small disjuncts. Also note that in this case pruning causes the overall error rate of the classifier to decrease from 6.9% to 5.3%.

4.1.2 The Effect of Pruning on Classifiers Induced from Thirty Data Sets

The performance of the classifiers induced from the thirty data sets, using C4.5 and Ripper with their default pruning strategies, is presented in Table 4.1 and Table 4.2, respectively. The induced classifiers are again placed into three categories, although in this case the patterns that were previously observed are not nearly as evident. This change can be measured by comparing the rank correlation coefficients. Pruning causes the Spearman rank correlation between accuracy and error concentration to drop from .86 to .68 for C4.5 from .80 to .47 for Ripper. Similarly, pruning causes the rank correlation between AUC and error concentration to drop from .91 to .75 for C4.5 and to drop from .94 to .61 for Ripper. Thus, with pruning, the values are only marginally correlated. We can also look at some specific examples. With pruning some classifiers continue to have low error rates but no longer have large error concentrations (e.g., *ocr*, *soybean-lg*, and, for C4.5 only, *ticket3*). In these cases pruning has caused the classification errors to be distributed much more uniformly throughout the disjuncts.

EC Rank	Dataset Name	Data Set Size	Largest Disjunct	% Errors at 10% Correct	% Errors at 20% Correct	% Correct at 50% Errors	Error Rate	AUC	Error Conc.
1	hypothyroid	3,771	2,732	90.7	90.7	0.7	0.5	.984	.818
2	ticket1	556	410	46.7	94.4	10.3	1.6	.936	.730
3	vote	435	221	68.7	74.7	2.9	5.3	.952	.712
4	breast-wisc	699	345	49.6	78.0	10.0	4.9	.943	.688
5	kr-vs-kp	3,196	669	35.4	62.5	15.6	0.6	.663	.658
6	splice-junction	3,175	479	41.6	45.1	25.9	4.2	.954	.566
7	crx	690	267	45.2	62.5	11.5	15.1	.856	.516
8	ticket2	556	442	48.1	55.0	12.8	4.9	.845	.474
9	weather	5,597	573	26.2	46.0	22.2	31.1	.931	.442
10	adult	21,280	5,018	36.6	53.2	17.6	14.1	.827	.424
11	german	1,000	313	29.6	46.8	21.9	28.4	.674	.404
12	soybean-large	682	61	48.0	57.3	14.4	8.2	N/A	.394
13	network2	3,826	1,685	30.8	48.2	21.2	22.2	.885	.362
14	ocr	2,688	1,350	40.4	46.4	34.3	2.7	.544	.348
15	market1	3,180	830	28.4	44.6	23.6	20.9	.757	.336
16	network1	3,577	1,470	24.4	43.4	27.2	22.4	.801	.318
17	ticket3	556	431	37.0	49.7	20.9	2.7	.847	.310
18	horse-colic	300	137	35.8	50.4	19.3	14.7	.721	.272
19	coding	20,000	415	17.2	31.6	34.9	27.7	.702	.216
20	sonar	208	50	15.1	28.0	34.6	28.4	.624	.202
21	heart-hungarian	293	132	19.9	37.7	31.8	21.4	.678	.198
22	hepatitis	155	89	24.2	46.3	26.3	18.2	.586	.168
23	liver	345	59	17.6	31.8	34.8	35.4	.661	.162
24	promoters	106	26	17.2	31.1	37.0	24.4	.676	.128
25	move	3,028	216	14.4	24.4	42.9	23.9	.649	.094
26	blackjack	15,000	3,053	16.9	29.7	44.7	27.6	.624	.092
27	labor	57	24	14.3	18.4	40.5	22.3	.568	.082
28	bridges	101	67	14.9	28.9	50.1	15.8	.669	.064
29	market2	11,000	426	12.2	23.9	44.7	45.1	.706	.060
30	bands	538	279	0.8	4.7	58.3	30.1	.545	-.184

Table 4.1: Error Concentration Results for C4.5 with Pruning

EC Rank	C4.5 Rank	Dataset Name	Data Set Size	Largest Disjunct	% Errors at 10% Correct	% Errors at 20% Correct	% Correct at 50% Errors	Error Rate	AUC	Error Conc.
1	1	hypothyroid	3,771	2,732	97.2	97.2	0.6	0.9	.973	.930
2	5	kr-vs-kp	3196	669	56.8	92.6	5.4	0.8	.999	.746
3	2	ticket1	556	410	41.5	95.0	11.9	1.6	.988	.740
4	6	splice-junction	3,175	552	46.9	75.4	10.7	5.8	.966	.690
5	3	vote	435	221	62.5	68.8	2.8	4.1	.976	.648
6	8	ticket2	556	405	73.3	74.6	7.8	4.5	.926	.574
7	17	ticket3	556	412	71.3	71.3	9.0	4.0	.944	.516
8	14	ocr	2,688	854	29.4	32.6	24.5	2.7	.967	.306
9	20	sonar	208	59	23.1	27.8	25.4	29.7	.725	.282
10	30	bands	538	118	22.1	39.5	24.0	26.0	.797	.218
11	9	weather	5,597	1,148	18.8	31.2	35.4	26.9	.851	.198
12	23	liver	345	69	13.6	33.2	34.7	32.1	.655	.146
13	12	soybean-large	682	66	17.8	26.6	47.4	9.8	N/A	.128
14	11	german	1,000	390	14.7	32.5	32.4	29.4	.665	.128
15	4	breast-wisc	699	370	14.4	39.2	31.4	4.4	.977	.124
16	15	market1	3,180	998	19.0	34.5	43.4	21.3	.839	.114
17	7	crx	690	272	16.4	31.9	39.1	15.1	.880	.108
18	13	network2	3,826	1,861	15.3	34.4	39.5	22.6	.749	.090
19	16	network1	3,577	1,765	16.0	34.4	42.0	23.3	.741	.090
20	18	horse-colic	300	141	13.8	20.5	36.6	15.7	.833	.086
21	21	hungarian-heart	293	138	17.9	29.3	42.6	18.8	.812	.072
22	19	coding	20,000	894	12.7	21.7	46.5	28.3	.767	.052
23	26	blackjack	15,000	4,893	16.8	22.1	45.3	28.1	.681	.040
24	22	hepatitis	155	93	25.5	28.3	57.2	22.3	.663	-.004
25	29	market2	11,000	2,457	7.7	17.7	50.2	40.9	.600	-.016
26	28	bridges	101	71	19.1	22.2	55.0	18.3	.735	-.024
27	25	move	3,028	320	10.9	19.5	63.1	24.1	.810	-.094
28	10	adult	21,280	9,293	9.8	29.5	67.9	15.2	.865	-.146
29	27	labor	57	25	0.0	3.6	70.9	18.2	.600	-.228
30	24	promoters	106	32	0.0	0.0	54.1	11.9	.928	-.324

Table 4.2: Error Concentration Results for Ripper with Pruning

The results in Tables 4.1 and 4.2, when compared to the results without pruning in Tables 3.1 and 3.2, show that pruning tends to reduce the error concentration of most classifiers. This comparison is shown graphically in Figure 4.3. In this figure, the x-axis represents the error concentration of the unpruned classifier while the y-axis represents the error concentration of the pruned classifier. Each point corresponds to the classifiers (unpruned and pruned) associated with a single data set. Since most of the points fall below the line Y=X, we conclude that for both C4.5 and Ripper, pruning, as expected, tends to reduce error concentration. However, the results in Figure 4.3 makes it clear that pruning has a more dramatic impact on the error concentration for classifiers induced using

Ripper than those induced using C4.5. Pruning causes the error concentration to decrease for 23 of the 30 data sets for C4.5 and for 26 of the 30 data sets for Ripper. More significant, however, is the magnitude of the changes in error concentration. On average, pruning causes the error concentration for classifiers induced using C4.5 to drop from .471 to .375, while the corresponding drop when using Ripper is from .445 to .206. These results indicate that the pruned classifiers produced by Ripper have the errors much less concentrated toward the small disjuncts than those produced by C4.5. Given that Ripper is generally known to produce very small rule sets, this larger decrease in error concentration is likely due to the fact that Ripper has a more aggressive pruning strategy than C4.5.

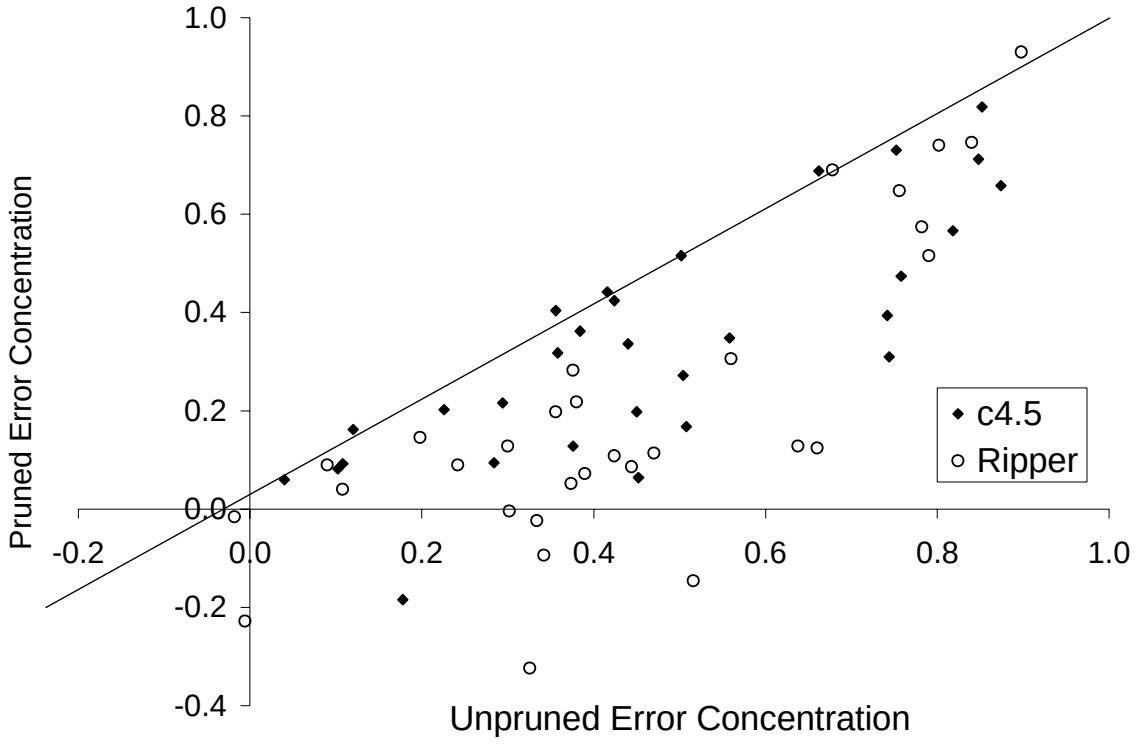

Figure 4.3: Effect of Pruning on Error Concentration

The results in Table 4.1 and Table 4.2 and in Figure 4.3 indicate that, even with pruning, the “problem with small disjuncts” is still quite evident for both C4.5 and Ripper.

For both learners the error concentration, averaged over the thirty data sets, is still decidedly positive. Furthermore, even with pruning both learners produce many classifiers with error concentrations greater than .500. However, it is certainly worth noting that the classifiers associated with seven of the data sets induced by Ripper with pruning have negative error concentrations.

Comparing the error concentration values for Ripper with and without pruning reveals one particularly interesting example. For the adult data set, pruning causes the error concentration to drop from .516 to -.146. This large change indicates that many error-prone small disjuncts are eliminated. This is supported by the fact that the size of the largest disjunct in the induced classifier changes from 1,488 without pruning to 9,293 with pruning. Thus, pruning seems to have an enormous effect on the classifier induced by Ripper from the adult data set. This may be explained by the fact that the adult data set is quite large (over 20,000 examples) and without pruning the learners will generate complex classifiers with an enormous number of rules.

For completeness, the effect that pruning has on error rate and AUC is shown graphically in Figures 4.4 and 4.5, respectively. Because most of the points in Figure 4.4 fall below the line Y=X, we conclude that pruning tends to reduce the error rate for both C4.5 and Ripper. However, the figure also makes it clear that pruning improves the performance of Ripper more than it improves the performance of C4.5. In particular, for C4.5 pruning causes the error rate to drop for 19 of the 30 data sets while for Ripper pruning causes the error rate to drop for 24 of the 30 data sets. Over the 30 data sets pruning causes C4.5's error rate to drop from 18.4% to 17.5% and Ripper's error rate to drop from 19.0% to 16.9%. Note that with pruning Ripper generally outperforms C4.5.

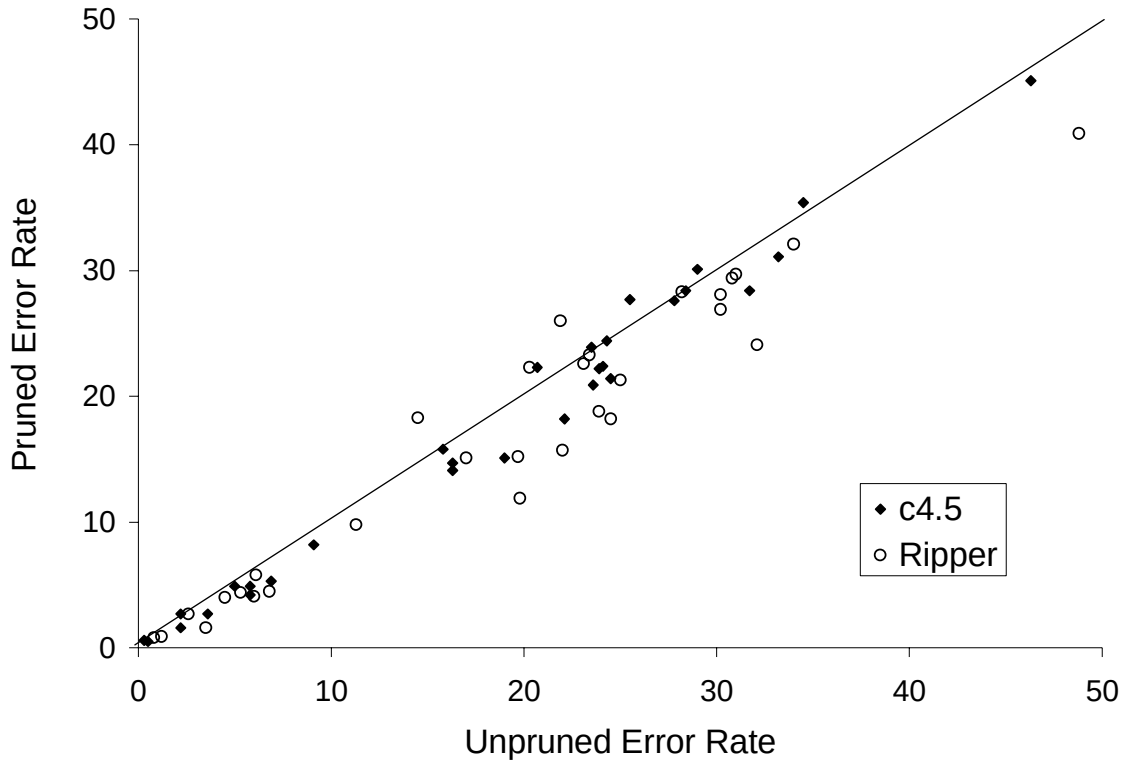

Figure 4.4: Effect of Pruning on Error Rate

The results in Figure 4.5 do not show any consistent pattern with respect to the effect of pruning on AUC. In some cases pruning reduces AUC and in other cases it increases the AUC. Pruning causes the average AUC for C4.5 to drop slightly, from .754 to .752, while pruning causes the average AUC value for Ripper to improve slightly, from .822 to .825. Thus, these results indicate that when AUC is used to measure performance, pruning is not very effective at improving classifier performance, for either C4.5 or Ripper. This result may warrant further study and suggests the need for pruning methods that are geared toward improving AUC.

The data underlying Figure 4.5 does indicate that Ripper consistently generates classifiers with better AUC values than C4.5—just as it did when pruning was not used. In this

case the classifiers induced by Ripper have higher AUC values than those induced by C4.5 in all but three cases.

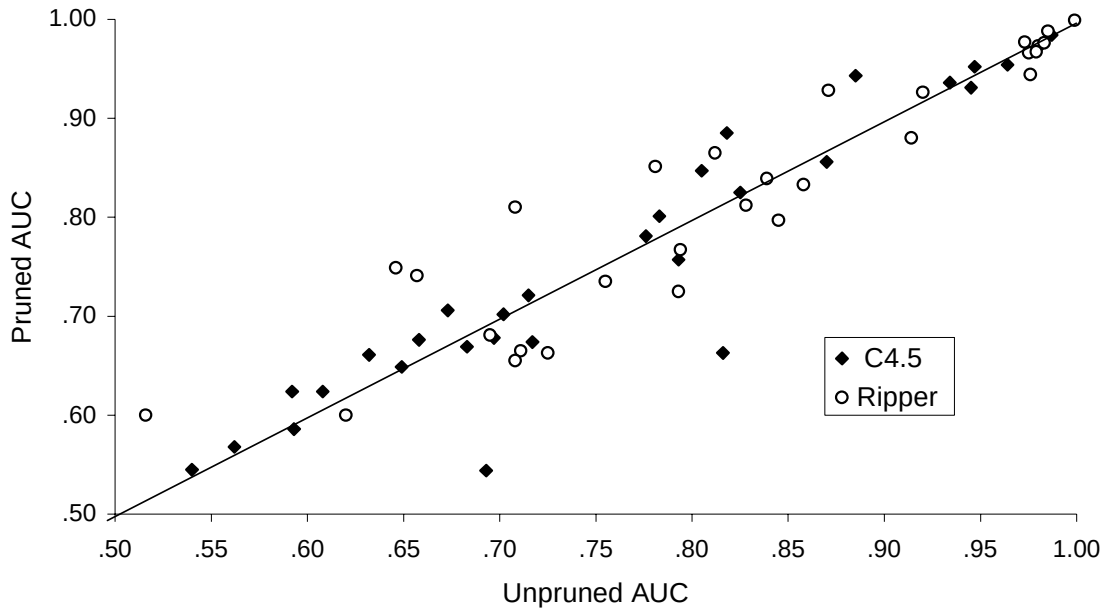

Figure 4.5: Effect of Pruning on AUC

4.1.3 Is Pruning More Effective for Classifiers with Large Error Concentrations?

Given that pruning tends to affect small disjuncts more than large disjuncts, a natural question that arises is whether pruning is more helpful when the errors in the unpruned classifier are most highly concentrated in the small disjuncts. Figure 4.6 addresses this by plotting the relative reduction in error rate due to pruning versus the error concentration rank of the unpruned classifier (those classifiers with the highest error concentration appear toward the left).

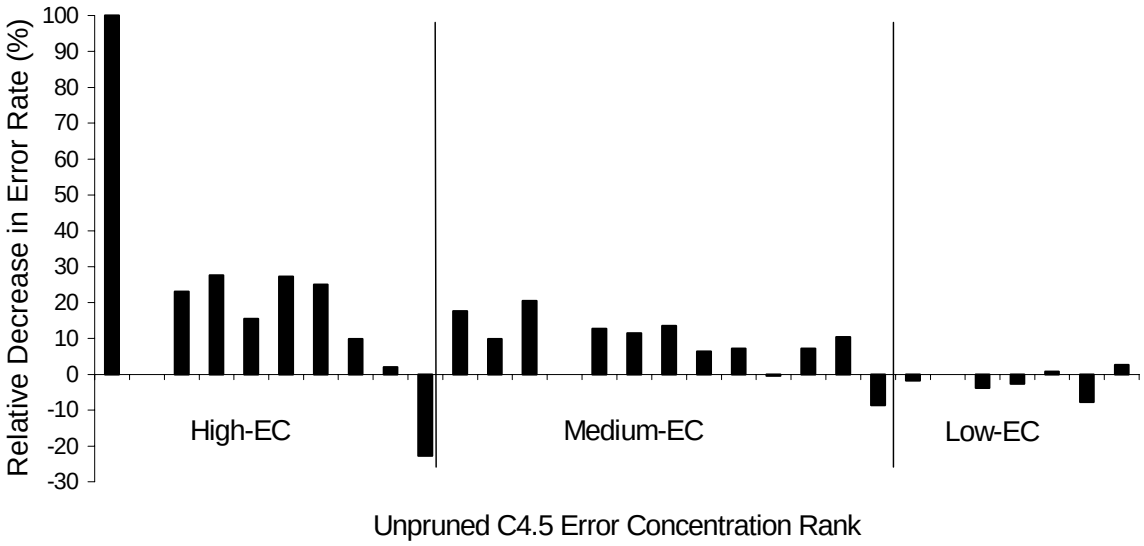

Figure 4.6: Improvement in Error Rate versus Error Concentration Rank

With few exceptions, the data sets with high and medium error concentrations show a decrease in error rate when pruning is used. However, most of the classifiers in the Low-EC category show an increase in error rate (the absolute change in error rate is fairly substantial for these classifiers because of the relatively high error rate without pruning). These results suggest that pruning is most beneficial when the errors are most highly concentrated in the small disjuncts—and may actually hurt when this is not the case.

4.1.4 Pruning as a Strategy for Addressing the Problem with Small Disjuncts

The results thus far show that pruning disproportionately eliminates small disjuncts and generally leads to a reduction in error rate. Hence, pruning can be considered a strategy for addressing the problem with small disjuncts. One can gauge the effectiveness of pruning as a strategy for addressing the problem with small disjuncts by comparing it to an "ideal" strategy that causes the error rate of the small disjuncts to equal the error rate of the other, larger, disjuncts.

Table 4.3 shows the average error rates of the classifiers induced by C4.5 for the thirty data sets, without pruning, with pruning, and with two variants of the idealized strategy. The error rates for the idealized strategies are computed by first identifying the smallest disjuncts that collectively cover 10% (20%) of the training examples; the error rate of the classifier is then recomputed assuming that the error rate of these disjuncts on the test set equals the error rate of the remaining disjuncts on the test set.

Strategy	No Pruning	Pruning	Idealized (10%)	Idealized (20%)
Average Error Rate	18.4%	17.5%	15.2%	13.5%
Relative Improvement		4.9%	17.4%	26.6%

Table 4.3: Comparison of Pruning to Idealized Strategy

The results in Table 4.3 show that the idealized strategy yields much more dramatic improvements in error rate than pruning, even when it is only applied to the disjuncts that cover 10% of the training examples. This indicates that there is still significant room for improvement in how small disjuncts are handled.

4.1.5 The Negative Impact of Pruning on Large Disjuncts

A consequence of pruning eliminating error-prone small disjuncts is that the large disjuncts will become more error prone. To investigate this further, we reverse our focus in this section and concentrate on the performance of the large disjuncts, with and without pruning. Our methodology is to form a classifier by starting with the largest disjunct and then progressively add smaller disjuncts, measuring the accuracy (i.e., precision) of the classifier at each step.

In addition to measuring the effect of pruning on large disjuncts, this process has an additional benefit. In many real-world situations one need not classify all examples, but

instead can be selective in choosing which examples to classify. As an example, consider a company that plans to initiate a direct-mail campaign promoting some of its products, in order to increase sales. In order to focus the campaign, the company would like to segment potential customers based on their buying preferences (e.g., book-buyers, dvd-buyers, etc.) and only promote the type of product (books, dvds, etc.) that the customer is most likely to purchase. The company has a classifier for segmenting customers, generated from past purchasing behavior and the results of past marketing campaigns. The company wants to target a subset of the customers for which it has purchasing information, in order to limit the budget of the direct-mail campaign. The company would like to contact only those customers for which it is most confident about the product that should be targeted.

Ideally, the company should select those rules (i.e., disjuncts) that are deemed "best" according to some combination of factors, such as training accuracy and the number of examples classified (since this affects the reliability of the estimated accuracy). Our results about the correlation between disjunct size and predictive accuracy suggests one possible strategy. This strategy, which we now explore, is to prefer classification rules that classify many training examples. Thus, those examples that are covered by these rules would be classified first. We also consider the effect that pruning has on this strategy.

Table 4.4 shows our results. The accuracy (i.e., precision) of the disjuncts in the classifier is shown both with and without pruning, at various points. For example, the first column after the data set name shows the performance of the largest disjuncts that collectively cover 10% of the training examples, when evaluated on the test data. Note that a

negative difference indicates that pruning leads to an improvement (i.e., a reduction) in error rate, while a positive difference indicates that pruning leads to an increase in error rate. Results are reported for classifiers with disjuncts that collectively cover 10%, 30%, 50%, 70% and 100% of the training examples.

Dataset	Error Rate with 10% covered			Error Rate with 30% covered			Error Rate with 50% covered			Error Rate with 70% covered			Error Rate with 100% covered		
Name	prune	none	Δ	prune	none	Δ	prune	none	Δ	prune	none	Δ	prune	none	Δ
kr-vs-kp	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.1	0.6	0.3	0.3
hypothyroid	0.1	0.3	-0.2	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.0	0.0	0.5	0.5	0.0
vote	3.1	0.0	3.1	1.0	0.0	1.0	0.9	0.0	0.9	2.3	0.7	1.6	5.3	6.9	-1.6
splice-junction	0.3	0.9	-0.6	0.2	0.3	-0.1	0.3	0.2	0.1	2.4	0.6	1.8	4.2	5.8	-1.6
ticket2	0.3	0.0	0.3	2.7	0.8	1.9	2.5	0.7	1.8	2.5	1.0	1.5	4.9	5.8	-0.9
ticket1	0.1	2.1	-1.9	0.3	0.6	-0.3	0.4	0.4	0.0	0.3	0.3	0.0	1.6	2.2	-0.5
ticket3	2.1	2.0	0.1	1.7	1.2	0.5	1.4	0.7	0.6	1.5	0.5	1.0	2.7	3.6	-0.9
soybean-large	1.5	0.0	1.5	5.4	1.0	4.4	5.3	1.6	3.7	4.7	1.3	3.5	8.2	9.1	-0.9
breast-wisc	1.5	1.1	0.4	1.0	1.0	0.0	0.6	0.6	0.0	1.0	1.4	-0.4	4.9	5.0	-0.1
ocr	1.5	1.8	-0.3	1.9	0.8	1.1	1.3	0.6	0.7	1.9	1.0	0.9	2.7	2.2	0.5
hepatitis	5.4	6.7	-1.3	15.0	2.2	12.9	15.0	9.1	5.9	12.8	12.1	0.6	18.2	22.1	-3.9
horse-colic	20.2	1.8	18.4	14.6	4.6	10.0	11.7	5.3	6.3	10.7	10.6	0.1	14.7	16.3	-1.7
crx	7.0	7.3	-0.3	7.9	6.5	1.4	6.3	7.3	-0.9	7.8	9.3	-1.6	15.1	19.0	-3.9
bridges	10.0	0.0	10.0	17.5	0.0	17.5	16.8	2.0	14.9	14.9	9.4	5.4	15.8	15.8	0.0
heart-hungarian	15.4	6.2	9.2	18.4	11.4	7.0	15.6	10.9	4.7	16.0	16.4	-0.4	21.4	24.5	-3.1
market1	16.6	2.2	14.4	12.2	7.8	4.4	12.7	12.1	0.6	14.5	15.9	-1.4	20.9	23.6	-2.6
adult	3.9	0.5	3.4	3.6	4.9	-1.3	8.9	8.1	0.8	8.3	10.6	-2.3	14.1	16.3	-2.2
weather	5.4	8.6	-3.2	10.6	14.0	-3.4	16.4	19.4	-3.1	22.7	24.6	-1.9	31.1	33.2	-2.1
network2	10.8	9.1	1.7	12.5	10.7	1.8	12.7	14.7	-2.0	15.1	17.2	-2.1	22.2	23.9	-1.8
promoters	10.2	19.3	-9.1	10.9	10.4	0.4	14.1	15.7	-1.6	19.6	16.8	2.8	24.4	24.3	0.1
network1	15.3	7.4	7.9	13.1	11.8	1.3	13.2	15.5	-2.3	16.7	17.3	-0.6	22.4	24.1	-1.7
german	10.0	4.9	5.1	11.1	12.5	-1.4	17.4	19.1	-1.8	20.4	25.7	-5.3	28.4	31.7	-3.3
coding	19.8	8.5	11.3	18.7	14.3	4.4	21.1	17.9	3.2	23.6	20.6	3.1	27.7	25.5	2.2
move	24.6	9.0	15.6	19.2	12.1	7.1	21.0	15.5	5.6	22.6	18.7	3.8	23.9	23.5	0.3
sonar	27.6	27.6	0.0	23.7	23.7	0.0	19.2	19.2	0.0	24.4	24.3	0.1	28.4	28.4	0.0
bands	13.1	0.0	13.1	34.3	16.3	18.0	34.1	25.0	9.1	33.8	26.6	7.2	30.1	29.0	1.1
liver	27.5	36.2	-8.8	32.4	28.1	4.3	28.0	30.1	-2.2	30.7	31.8	-1.2	35.4	34.5	0.9
blackjack	25.3	26.1	-0.8	25.1	25.8	-0.8	24.8	26.7	-1.9	26.1	24.4	1.7	27.6	27.8	-0.2
labor	25.0	25.0	0.0	17.5	24.8	-7.3	23.6	20.3	3.2	24.4	17.5	6.9	22.3	20.7	1.6
market2	44.1	45.5	-1.4	43.1	44.3	-1.2	42.5	44.2	-1.7	43.3	45.3	-2.0	45.1	46.3	-1.2
Average	**11.6**	**8.7**	**2.9**	**12.5**	**9.7**	**2.8**	**12.9**	**11.4**	**1.5**	**14.2**	**13.4**	**0.8**	**17.5**	**18.4**	**-0.9**

Table 4.4: Effect of Pruning when Classifier Generated using Largest Disjuncts

The last row in Table 4.4 shows the error rates averaged over the thirty data sets. These results clearly show that, over the thirty data sets, pruning only helps for the last column—when all disjuncts are included in the evaluated classifier. Note that these re-

sults, which correspond to the accuracy results presented earlier, are typically the only results that are reported. The results from the last row of Table 4.4 are displayed graphically in Figure 4.7, which plots the error rates, with and without pruning, averaged over the thirty data sets. Note, however, that unlike the results in Table 4.4, Figure 4.7 shows classifier performance at each 10% increment.

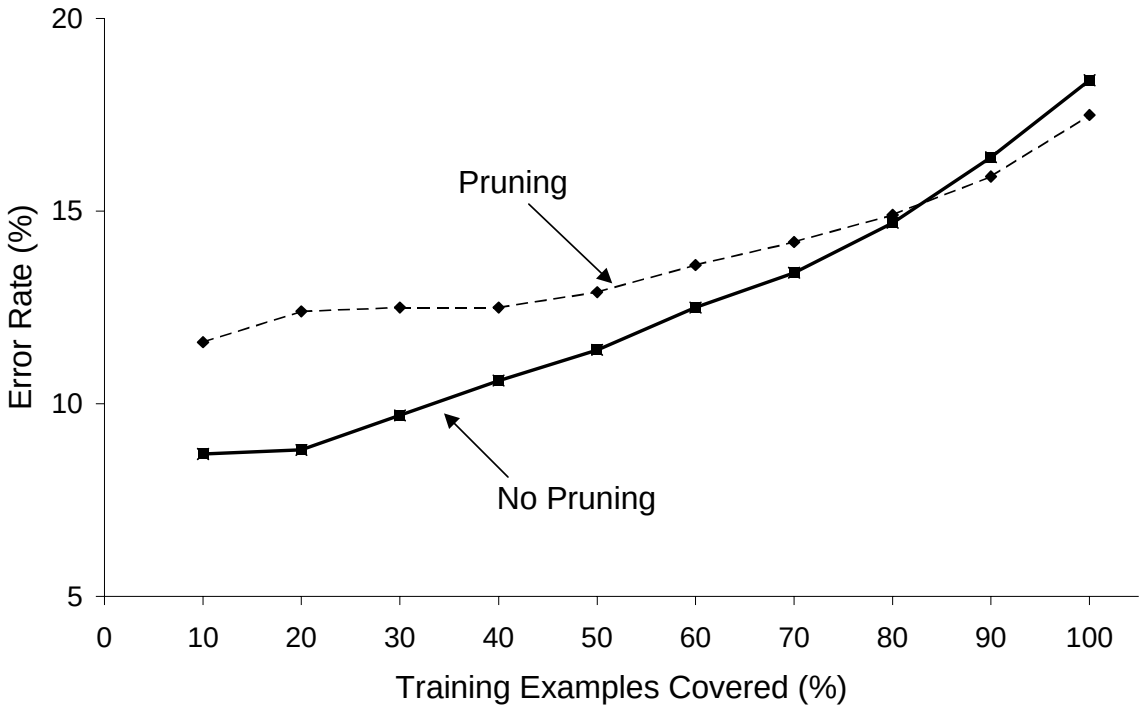

Figure 4.7: The Impact of Pruning on Large Disjuncts

Figure 4.7 shows that pruning degrades the performance of the highly accurate large disjuncts and that if disjunct size is used to select the best rules, then pruning will degrade classifier performance in most situations when all examples need not be classified. These results confirm the hypothesis that when pruning eliminates some small disjuncts, the emancipated examples cause the error rate of the more accurate large disjuncts to decrease. The overall error rate is reduced only because the error rate associated with the

emancipated examples is lower than their original error rate. Thus, pruning redistributes the errors such that the errors are more uniformly distributed than without pruning. This net decrease in accuracy is only overcome, on average, when the classifier includes disjuncts that cover at least 80% of the training examples.

4.2 Training-Set Size

The amount of training data is known to affect the structure of a classifier and its performance. For example, providing additional training data typically leads to a more complex model (e.g., more leaves, rules, etc.) with improved classification performance. In this section we analyze the effect that training-set size has on small disjuncts and error concentration.

This section, like the previous section, begins with a simple example based on the *vote* data set. The distribution of errors for the *vote* data set is displayed (again) in Figure 4.8. Figure 4.9 shows the distribution of errors when the training set is limited to use only 10% of the total data. Since the results in Figure 4.8 are based on 10-fold cross validation, which uses 90% of the data for training, the classifier described in Figure 4.9 uses one-ninth the training data as the one described in Figure 4.8. Note that the size of the bins, and consequently the scale of the x-axis, has been reduced in Figure 4.9.

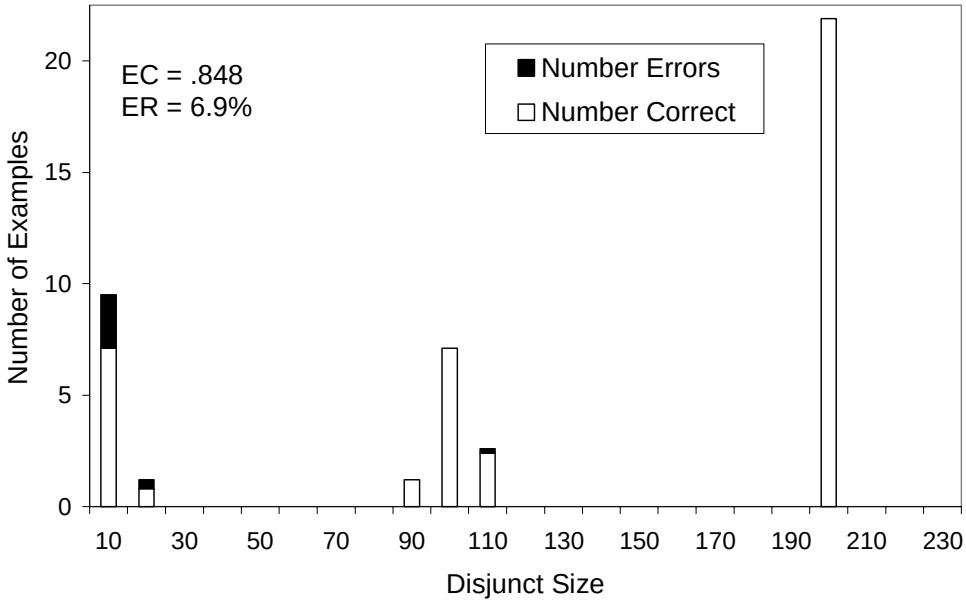

Figure 4.8: Distribution of Errors Using the Full Training Set

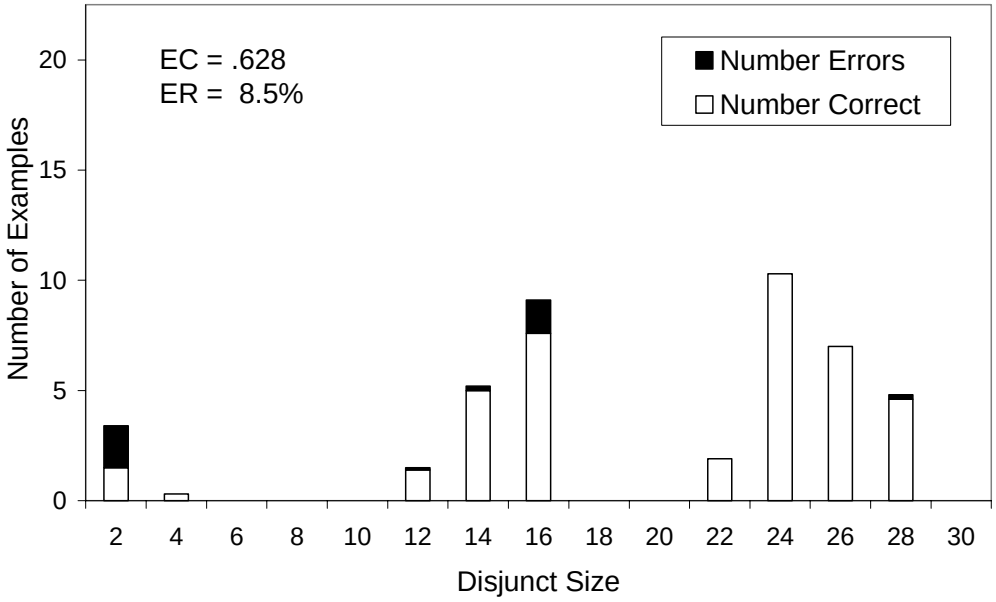

Figure 4.9: Distribution of Errors using One-Ninth the Training Data

Comparing the relative distribution of errors between Figure 4.8 and Figure 4.9 shows that errors are more concentrated toward the smaller disjuncts in Figure 4.8, which has a higher error concentration (.848 vs. .628). This indicates that increasing the amount of training data increases the degree to which the errors are concentrated toward the small disjuncts. Like the results in Figure 4.8, the results in Figure 4.9 show that there are three

groupings of disjuncts, which one might be tempted to refer to as small, medium, and large disjuncts. The size of the disjuncts within each group differs between the two figures, due to the different number of training examples used to generate each classifier (note the change in scale of the x-axis).

Because error concentration is a relative measure, it is meaningful to compare the error concentrations for classifiers induced using different training-set sizes. Table 4.5 shows the error rate and error concentration for the classifiers induced from each of the thirty data sets using three different training set sizes.

	Amount of Total Data Used for Training						Change from	
	10%		50%		90%		10% to 90%	
Data Set Name	ER	EC	ER	EC	ER	EC	ER	EC
kr-vs-kp	3.9	.742	0.7	.884	0.3	.874	-3.6	.132
hypothyroid	1.3	.910	0.6	.838	0.5	.852	-0.8	-.058
vote	9.0	.626	6.7	.762	6.9	.848	-2.1	.222
splice-junction	8.5	.760	6.3	.806	5.8	.818	-2.7	.058
ticket2	7.0	.364	5.7	.788	5.8	.758	-1.2	.394
ticket1	2.9	.476	3.2	.852	2.2	.752	-0.7	.276
ticket3	9.5	.672	4.1	.512	3.6	.744	-5.9	.072
soybean-large	31.9	.484	13.8	.660	9.1	.742	-22.8	.258
breast-wisc	9.2	.366	5.4	.650	5.0	.662	-4.2	.296
ocr	8.9	.506	2.9	.502	2.2	.558	-6.7	.052
hepatitis	22.2	.318	22.5	.526	22.1	.508	-0.1	.190
horse-colic	23.3	.452	18.7	.534	16.3	.504	-7.0	.052
crx	20.6	.460	19.1	.426	19.0	.502	-1.6	.042
bridges	16.8	.100	14.6	.270	15.8	.452	-1.0	.352
heart-hungarian	23.7	.216	22.1	.416	24.5	.450	0.8	.234
market1	26.9	.322	23.9	.422	23.6	.440	-3.3	.118
adult	18.6	.486	17.2	.452	16.3	.424	-2.3	-.062
weather	34.0	.340	32.7	.380	33.2	.416	-0.8	.076
network2	27.8	.354	24.9	.342	23.9	.384	-3.9	.030
promoters	36.0	.108	22.4	.206	24.3	.376	-11.7	.268
network1	28.6	.314	25.1	.354	24.1	.358	-4.5	.044
german	34.3	.248	33.3	.334	31.7	.356	-2.6	.108
coding	38.4	.214	30.6	.280	25.5	.294	-12.9	.080
move	33.7	.158	25.9	.268	23.5	.284	-10.2	.126
sonar	40.4	.028	27.3	.292	28.4	.226	-12.0	.198
bands	36.8	.100	30.7	.152	29.0	.178	-7.8	.078
liver	40.5	.030	36.4	.054	34.5	.120	-6.0	.090
blackjack	29.4	.100	27.9	.094	27.8	.108	-1.6	.008
labor	30.3	.114	17.0	.044	20.7	.102	-9.6	-.012
market2	47.3	.032	45.7	.028	46.3	.040	-1.0	.008
Average	23.4	.347	18.9	.438	18.4	.471	-5.0	.124

Table 4.5: The Effect of Training-Set Size on Error Concentration

The last two columns in Table 4.5 highlight the impact that training-set size has on error rate and error concentration. The values in these two columns show how the error rate and error concentration change when the training set size is increased by a factor of nine. As expected, the error rate tends to decrease with additional training data. The error concentration, consistent with the results associated with the vote data set, shows a consistent increase—for 27 of the 30 data sets the error concentration increases when the amount of training data is increased by a factor of nine.

The observed change in error concentration can be explained. First note that the largest disjunct in Figure 4.8 does not cover a single error and that the medium-sized disjuncts, with sizes between 80 and 109, cover only a few errors. Their counterparts in Figure 4.9, with size between 20 and 27 and 10 to 15, have higher error rates. Thus, in this case an increase in training data leads to more accurate large disjuncts and a higher error concentration.

One potential explanation for the increase in error concentration has to do with the potential inability of the learner to perfectly express the target concept. Consider the example in Figure 3.1 of Chapter 3, where the target concept is a square rotated forty-five degree with respect to the x-axis. The large disjunct covers most of the target concept—the part that can be learned using axis-parallel cuts of the space, while the small disjuncts are an attempt to learn the decision boundaries that are not expressible with axis-parallel decision boundaries. As more training data becomes available, the error rate of the large disjunct should decrease, since the decision boundaries associated with the portion of the concept that can be learned using axis-parallel cuts will move closer to the boundaries of the target concept (in Figure 3.1 they will move inward so they do not extend as far out-

side of the target concept). However, even with more data the classifier cannot learn the decision boundary of the target concept that is not axis-parallel, and hence there will still be errors associated with the small disjuncts. In fact, the additional training data may cause more, error prone, small disjuncts to be formed, in an attempt to better approximate this surface. The net effect will be that the error concentration will increase.

Next consider the second scenario, shown previously in Figure 3.2, where there are large subconcepts and small subconcepts within the target concept. Additional training data will generally improve the ability to learn the large subconcepts. It will also tend to improve the ability to learn the small subconcepts, but the increased amount of training data may now make it possible to sample points within some of these small subconcepts for the first time. This will lead to new small disjuncts, which will be error prone due to the small number of examples that are available for learning (i.e., the learned decision boundary is likely to deviate significantly from the true decision boundary). In addition, the increased amount of training data may also cause some small disjuncts to be formed erroneously, due to noise and other similar real-world issues. These effects will also tend to increase the error concentration of the learned concept.

In this section we show that additional training data reduces the error rate of induced classifiers and increases their error concentration. These results help to explain the pattern, described in Chapter 3, that classifiers with low error rates tend to have higher error concentrations that those with high error rates. That is, if we imagine that additional training data were made available to those data sets where the associated classifier has a high error rate, we would expect the error rate to decline and the error concentration to increase. This would tend to move classifiers into the High-EC category, which includes

classifiers with relatively low error rates. Thus, to a large extent, the pattern that was established in Chapter 3 between classifier performance and error concentration reflects the degree to which a concept has been learned—concepts that have been well-learned tend to have very large disjuncts which are extremely accurate and hence have low error concentrations.

4.3 Noise

Noise plays an important role in classifier learning. Both the structure and performance of a classifier is affected by noisy data. In particular, noisy data may cause many erroneous small disjuncts to be induced. Danyluk and Provost (1993) speculated that the classifiers they induced from (systematic) noisy data performed poorly because of an inability to distinguish between these erroneous consistencies and correct ones. Weiss (1995) and Weiss and Hirsh (1998) explored this hypothesis using, respectively, two artificial data sets and two real-world data sets and showed that noise can make small subconcepts within the target concept (i.e., the so-called true exceptions) difficult to learn.

The research presented in this section further investigates the role of noise in learning, and, in particular, shows how noisy data affects classifiers and the distribution of errors within these classifiers. The experiments described in this section involve applying random class noise and random attribute noise to the data. The following experimental scenarios are explored:

Scenario 1: Random class noise is applied to the training data

Scenario 2: Random attribute noise is applied to the training data

Scenario 3: Random attribute noise is applied to both the training and test data

Class noise is only applied to the training set since the uncorrupted class label in the test set is required to properly measure classifier performance. The second scenario, in which random attribute noise is applied only to the training set, permits us to measure the sensitivity of the learner to noise (if attribute noise were applied to the test set then even if the correct concept were learned there would be classification errors). The third scenario, in which attribute noise is applied to both the training and test set, corresponds to the real-world situation where errors in measurement affect all examples.

Noise is defined as follows. A level of n% random class noise means that for n% of the examples the class label is replaced by a randomly selected class value (possibly the same as the original value). Attribute noise is defined similarly, except that for numerical attributes a random value is selected between the minimum and maximum values that occur within the data set. Note that only when the noise level reaches 100% is all information contained within the original data lost.

The vote data set is used to illustrate the effect that noise has on the distribution of examples, by disjunct size. The results are shown in Figure 4.10, with the graphs in the left column corresponding to the case when there is no pruning and the graphs in the right column corresponding to the case when pruning is used. Figures 4.10a and 4.10b, which have been displayed previously, show the results without any noise and are provided for comparison purposes. Figures 4.10c and 4.10d correspond to the case where 10% attribute noise is applied to the training data and Figures 4.10e and 4.10f correspond to the case where 10% class noise is applied to the training data.

No Pruning **With Pruning**

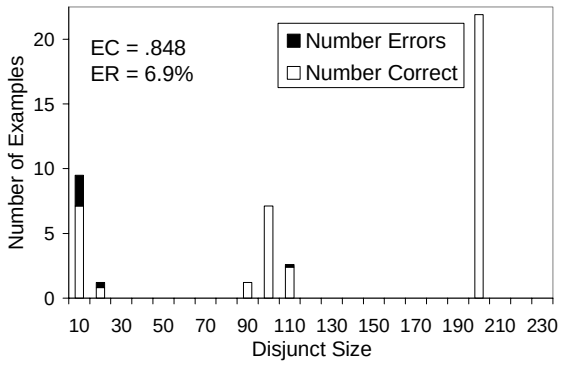

Figure 4.10a: No Noise

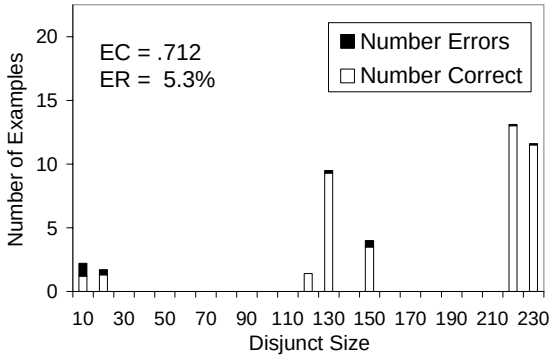

Figure 4.10b: No Noise

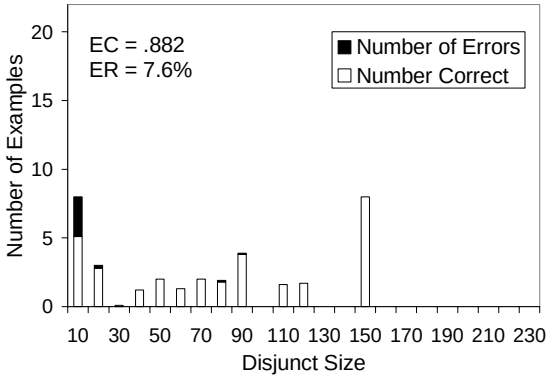

Figure 4.10c: 10% Attribute Noise

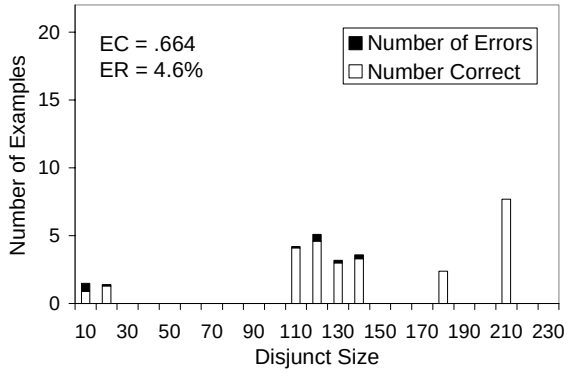

Figure 4.10d: 10% Attribute Noise

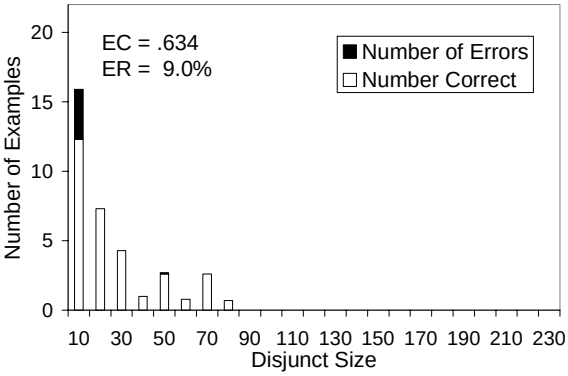

Figure 4.10e: 10% Class Noise

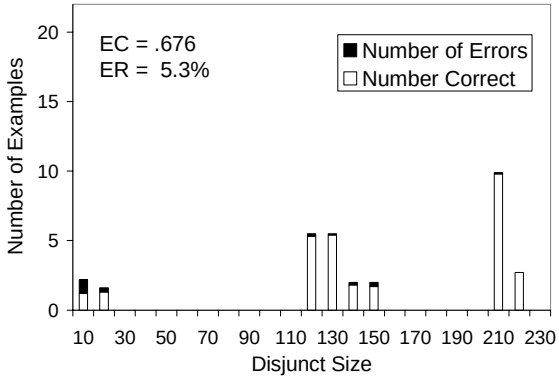

Figure 4.10f: 10% Class Noise

Figure 4.10: Effect of Noise on the Distribution of Errors

A comparison of Figure 4.10a with Figures 4.10c and 4.10e shows that without pruning both attribute and class noise cause more test examples to be covered by small disjuncts and less to be covered by the large disjuncts. This shift is much more dramatic for class noise than for attribute noise, and, as is indicated by the error rates, the class noise has a greater impact on classifier performance. The underlying data indicate that this shift occurs because noisy data causes many more small disjuncts to be formed. This comparison also shows that the error concentration remains fairly stable when attribute noise is added but decreases significantly when class noise is added.

Pruning reduces the shift in the distribution of correctly and incorrectly classified examples just described. This can be seen by comparing the classifiers induced from noisy data, without and then with pruning. Specifically, the distributions in Figures 4.10d and 4.10f are much more similar to the distribution in Figure 4.10b than the distributions in Figures 4.10c and 4.10e are to the distribution in Figure 4.10a. A comparison of the error rates for classifiers with and without pruning also shows that pruning is able to combat the effect of noise on the ability of the classifier to learn the concept. Surprisingly, when pruning is used, classifier accuracy for the vote data set actually improves when 10% attribute noise is added—the error rate decreases from 5.3% to 4.6%. This phenomenon, which is discussed in more detail shortly, is actually observed for many of the thirty data sets, but only when low (e.g., 10%) levels of attribute noise are added. The error concentration results also indicate that even with pruning, noise causes the errors to be distributed more uniformly throughout the disjuncts than when no noise is applied.

The results presented in the remainder of this section are based on averages over twenty-seven of the thirty data sets listed in Table 2.3. [1] The next three figures show, respectively, how noise affects the number of leaves, the error rate and the error concentration of the induced classifiers. Measurements are taken at the following noise levels: 0%, 5%, 10%, 20%, 30%, 40%, and 50%. The curves in these figures are labeled to identify the type of noise that is applied, whether it is applied to the training set or training and test set and whether pruning is used. The labels are interpreted as follows: the "Class" and "Attribute" prefix indicate the type of noise, the "-Both" term, if included, indicates that the noise is applied to the training and test sets rather than to just the training set, and the "-Prune" suffix is used to indicate that the results are with pruning.

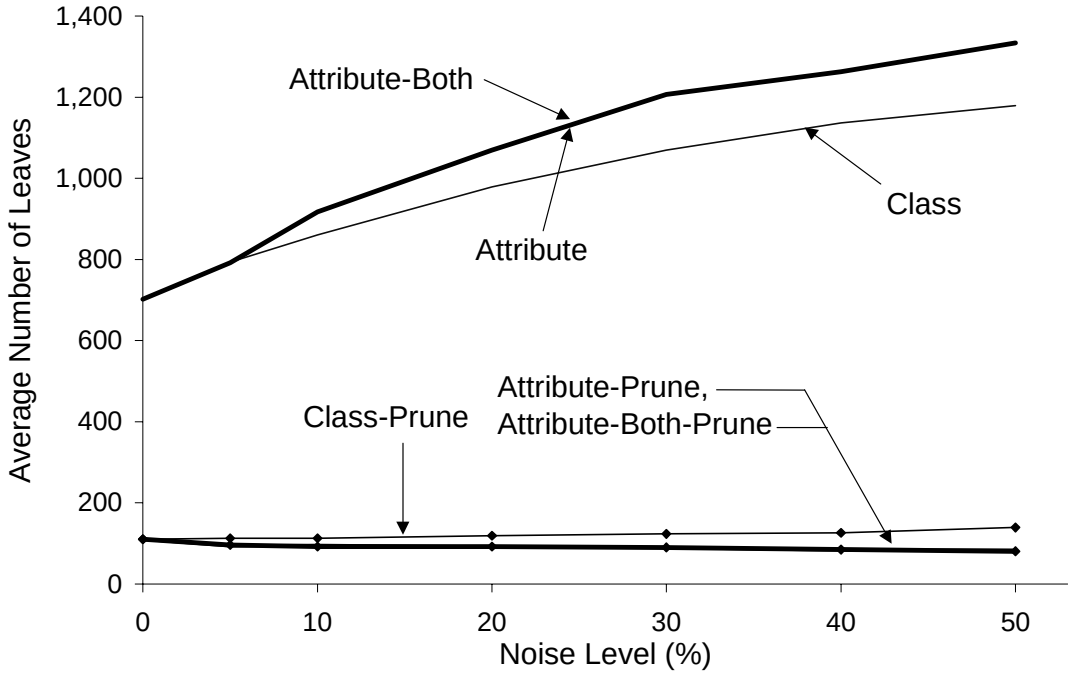

Figure 4.11: The Effect of Noise on Classifier Complexity

[1] The *coding*, *ocr* and *bands* data sets were excluded because of resource limitations in the program used to apply the noise model to the data. These limitations had to do with the maximum number of allowed attributes and attribute values.

The results in Figure 4.11 show that without pruning the number of leaves in the induced decision trees increase dramatically with increasing levels of noise, but that pruning effectively eliminates this increase. This indicates that pruning is able to prevent noise from generating large numbers of additional small disjuncts.

The effect that noise has on error rate is shown in Figure 4.12. The error rate of the induced classifiers increases with increasing levels of noise, with one exception. When attribute noise is applied to only the training data and pruning is used, the error rate decreases slightly from 17.7% with 5% noise to 17.5% with 10% noise. This decrease is no anomaly, since it occurs for most of the data sets. This decrease in error rate may be due to the fact that attribute noise leads to more small disjuncts and consequently more aggressive pruning, which turns out to be beneficial. A comparison of the results in Figure 4.12 for class noise with and without pruning (Class-Prune and Class) with the results for attribute noise with and without pruning (Attribute and Attribute-Prune) shows that pruning is far more effective at handling class noise than attribute noise.

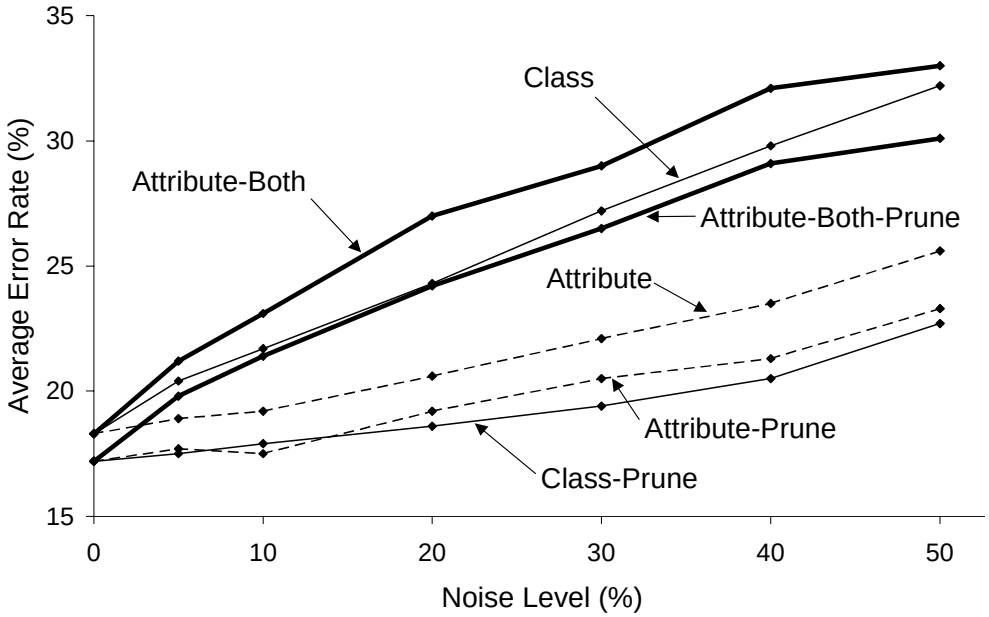

Figure 4.12: The Effect of Noise on Error Rate

Figure 4.13 shows the effect of noise on error concentration. When pruning is not employed, increasing the amount of noise leads to a decrease in error concentration, indicating that errors become more uniformly distributed based on disjunct size. This helps explain why we find a high-EC group of classifiers with good classifier performance and a medium-EC group of classifiers with worse classifier performance: adding noise to classifiers in the former increases their error rate and decreases their error concentration, making them look more like classifiers in the latter group. The results in Figure 4.13 also show, however, that when there is noise only in the training set (Class-Prune, Attribute-Prune), pruning causes the error concentration to remain relatively constant. This is the same thing that was observed for the vote data set in Figures 4.10d and 4.10e—pruning prevents noisy data from breaking down the highly accurate large disjuncts (i.e., pruning prunes back the newly formed small disjuncts, reconstituting the large disjuncts).

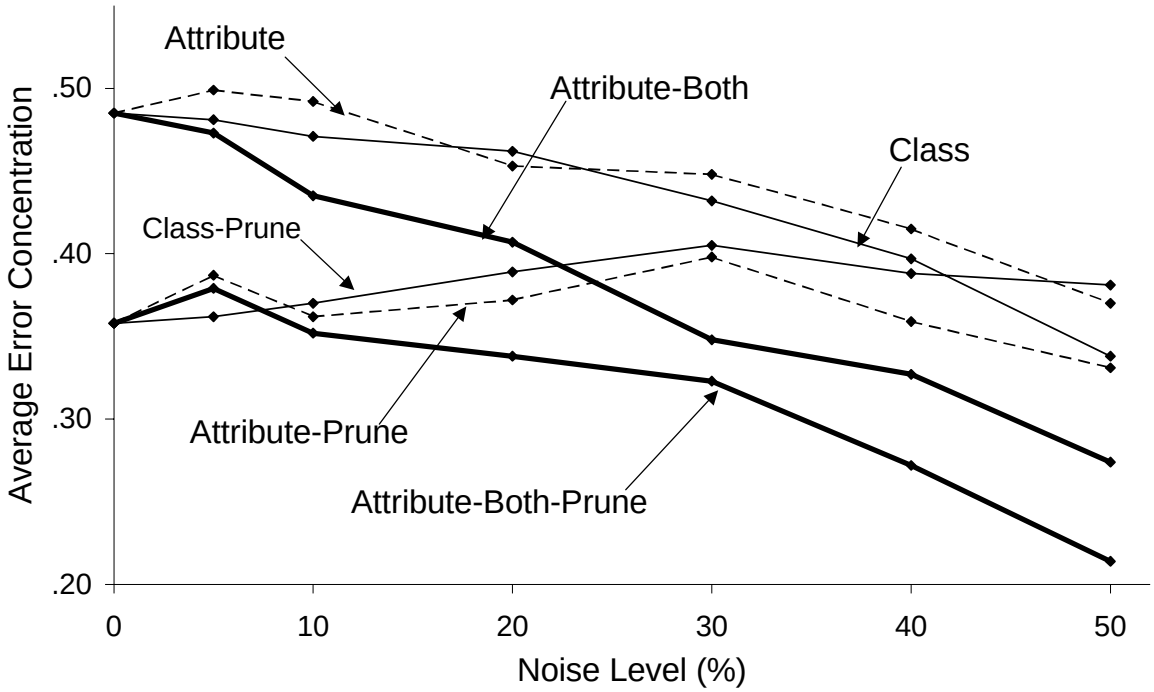

Figure 4.13: The Effect of Noise on Error Concentration

The results in this section demonstrate that pruning enables the learner to combat noisy training data. Specifically, pruning removes many of the disjuncts that are caused by the noise (Figure 4.11) and this results in a much smaller increase in error rate than if pruning were not employed (Figure 4.12). Because pruning eliminates many of the erroneous small disjuncts, the errors are not nearly as concentrated in the small disjuncts (Figure 4.13). We believe that the increase in error rate that comes from noisy training data when pruning is employed is at least partly due to the inability of the learner to distinguish between small subconcepts in the target concept and apparent subconcepts that are due to noise.

The detailed results associated with the individual data sets show that for class noise there is a trend for data sets with high error concentrations to experience a large increase in error rate from class noise and for those with low error concentrations to experience a much smaller increase in error rate. The most striking observation is that the induced classifiers with very low error concentrations are extremely tolerant of class noise, while none of the other classifiers exhibit this property. For example, the blackjack and labor data sets, both of which have low error concentrations (.108 and .102, respectively using C4.5), are so tolerant of noise that when 50% random class noise is added to the training set, the error rate on the induced classifier on the test data increases by less than 1%.

These results are explained if noise makes learning difficult because of an inability to distinguish between small subconcepts of the target concept and noisy data. Classifiers with a high error concentration already show an inability to properly learn the small subconcepts in the target concept (based on the assumption that small disjuncts correspond to small subconcepts)—the addition of noise simply worsens the situation. Concepts with

very large subconcepts that can be learned well without noise tend to form classifiers with highly accurate large disjuncts. These target concepts should be less susceptible to class noise. To see this, observe that corrupting the class labels for a few examples that are covered by a very large disjunct is unlikely to change the class label learned for that disjunct.

4.4 Class Imbalance

A data set exhibits class imbalance if the number of examples belonging to each class is not identical. This section investigates the relationship between class imbalance and error concentration by altering the class distribution of a group of data sets and then measuring the impact that this has on the error concentration of the induced classifiers. The connection between class imbalance and error concentration is suggested by some of the results from the study of class distribution in Chapter 6. For simplicity, we look at only two class distributions for each data set: the naturally occurring class distribution and a perfectly balanced class distribution, in which each class is represented in equal proportions. By comparing the error concentrations for these two class distributions, one can also determine how much of the "problem with small disjuncts" is due to class imbalance in the data set.

The experiments described in this section require that the class distribution of the data sets be modified. Thus, these experiments utilize the experimental methodology associated with the class distribution experiments. This means that the twenty-six data sets described in Table 2.4 are used in this section, rather than the thirty data sets that were employed in earlier sections. The balanced version of the data set is formed using the meth-

odology described later in Section 6.5.1; however, in this case the distribution of the test set is also altered to form a balanced distribution.

Figure 4.14 shows the error concentration for the classifiers induced by C4.5, using the natural and balanced versions of the data sets listed in Table 2.4. Since the error concentrations are all greater than zero when there is no class imbalance, we conclude that even with a balanced data set, errors tend to be concentrated toward the smaller disjuncts. However, by comparing the error concentrations associated with the classifiers induced from the balanced and natural class distributions, we see that when there is class imbalance, with few exceptions, the error concentration increases. The differences tend to be larger when the data set has greater class imbalance (the leftmost data set has the most natural class imbalance and the class imbalance decreases from left to right).

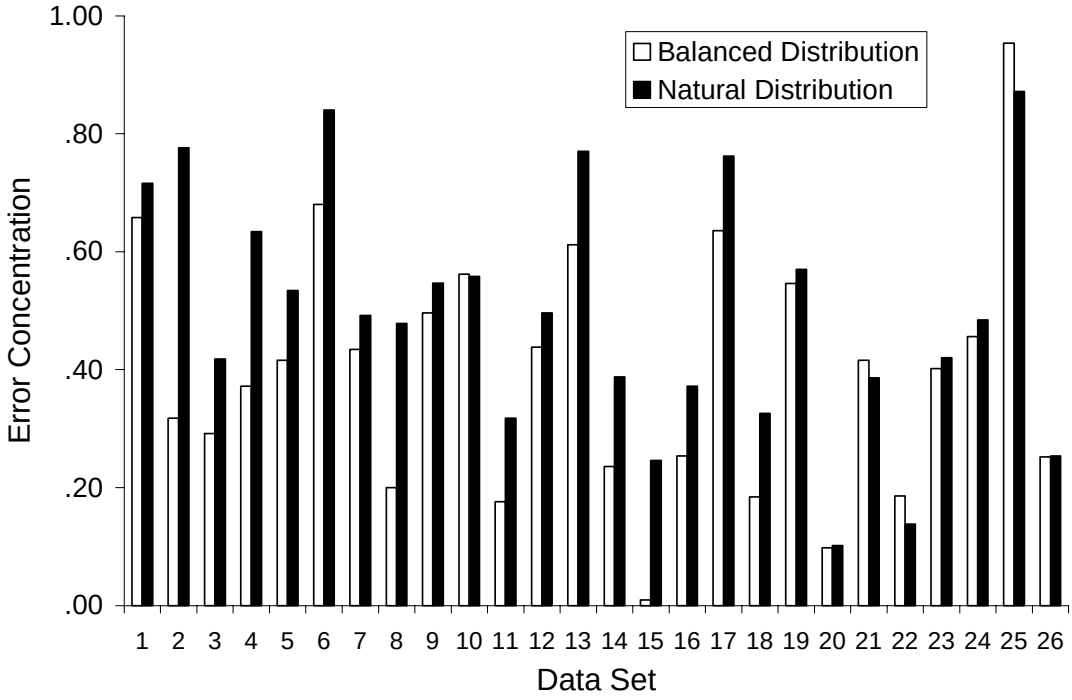

Figure 4.14: The Effect of Class Distribution on Error Concentration

The average error concentration of the classifiers induced from the balanced versions of the data sets is .396 while the average error concentration associated with the naturally occurring class distributions is .496. This corresponds to a 20% reduction in error concentration when class imbalance is removed. If we restrict our attention to the first 18 data sets that contain at most 30% minority-class examples, then the differences in error concentration are 28% (.387 for the balanced data sets versus .537 for the data sets with the natural class distributions).

We therefore conclude that for data sets with class imbalance, part of the reason why small disjuncts have a higher error rate than the large disjuncts is due to class imbalance. This is the first empirical evidence that class imbalance is partly responsible for the problem with small disjuncts. A complete explanation for this behavior requires a better understanding of class distribution and therefore is deferred until Chapter 6. However, a brief explanation is provided here. As we shall see in Chapter 6, minority-class predictions (i.e., minority-labeled leaves, rules, etc.) tend to be formed from fewer training examples than majority-class predictions and hence are more often associated with small disjuncts. Because of the test-distribution effect to be described in Chapter 6, minority-class examples are more difficult to classify than majority-class examples. Therefore, part of the reason why small disjuncts have a higher error rate than large disjuncts is that they are disproportionately labeled with the minority class—the class harder to classify correctly.

4.5 Summary

This chapter investigated how pruning, training-set size, noise, and class imbalance affect error concentration, and, more generally, affect learning. In particular, a great deal of

attention was devoted to pruning. The results in this chapter show that pruning disproportionately eliminates small disjuncts. This causes the errors to be distributed much more uniformly throughout the disjuncts. As a consequence, the results in this chapter show that while pruning generally leads to an overall decrease in error rate, it leads to an increase in error rate of the larger disjuncts, which tend to be the most accurate. This may lead to *poorer* classification performance when it isn't necessary to classify all examples. Pruning was also shown to be ineffective at improving classifier performance when the unpruned classifier has a low error concentration (i.e., when the errors are spread somewhat uniformly across the disjuncts). Finally, pruning was also evaluated as a method for addressing the problem with small disjuncts and shown to be only partly effective.

The analysis of training-set size shows that an increase in the amount of training data almost always leads to an increase in error concentration. This change occurs because as more training data is made available, the large disjuncts, which may cover the areas in the target concept that are expressible using axis-parallel cuts in the instance space, can be learned more accurately. In contrast, small disjuncts, which may be used to approximate the portions of the target concept that cannot be expressed using axis-parallel cuts (and generally lie close to the decision boundary), will still contain some errors. Furthermore, the additional training data may cause some small subconcepts to be sampled for the first time, resulting in small disjuncts. These disjuncts will tend to be error prone because with few training examples, it will be difficult to accurately determine the correct boundaries of the subconcept.

The section on noise shows that noisy training data leads to the formation of many additional, erroneous, small disjuncts, and the breakdown of many of the highly accurate large disjuncts. Pruning dramatically reduces the number of small disjuncts that are formed due to the noisy training data, which proves to be quite effective; pruning greatly diminishes the reduction in classifier accuracy that accompanies noise. The results also indicate that classifiers with high error concentrations are much more sensitive to class noise than those with low error concentrations. This suggests that the increase in error rate that comes from noisy training data when pruning is employed is at least partly due to the inability of the learner to distinguish between small subconcepts in the target concept and noisy data that appear similar, or identical to, such subconcepts.

Class imbalance is shown to affect error concentration. Increasing the amount of class imbalance, or skew, is shown to increase the error concentration of the induced classifier. This directly ties some of the research into small disjuncts with the research into class distribution. Although a complete explanation is provided in Chapter 6, the reason that class imbalance affects error concentration is that minority-class examples are more difficult to classify than majority-class examples, and small disjuncts are disproportionately labeled with the minority class. Thus class imbalance, and in particular the minority class, is partly responsible for the problem with small disjuncts.

Chapter 5

Research Related to Small Disjuncts

There have been a number of research papers that specifically address the topic of small disjuncts. This research is summarized in this section. Rather than describing the research chronologically, research into small disjuncts is divided into three areas and the research associated with each area is described in separate subsections. Research into small disjuncts can be placed into the following categories, based on the purpose of the research (the research in this thesis focuses on the first two categories):

- Research to *measure* the role and impact of small disjuncts on learning.
- Research to provide a better *understanding* of small disjuncts (such as why they are more error prone than large disjuncts).
- Research into how to *address* the problem of small disjuncts by building better learners.

5.1 Measuring Small Disjuncts

Prior research into small disjuncts did not focus on providing a thorough empirical analysis of the role small disjuncts play in learning. As stated earlier in this thesis, previous research efforts fully analyze only one or two data sets. In particular, Holte et al. (1989) analyze two data sets, Ali and Pazzani (1992) one data set, Danyluk and Provost (1993)

one data set, Weiss (1995) two data sets, Weiss and Hirsh (1998) two data sets, and Carvalho and Freitas (2000) two data sets.

Of these research studies, only the study by Danyluk and Provost (1993) was focused primarily on assessing the impact of small disjuncts on learning. In this research, one domain was studied in detail, to show that when inductive learning is used to form a knowledge base of rules for diagnosing telecommunication problems, small disjuncts are necessary for high accuracy even though they have a relatively high error rate. Chapter 3 extended this research by measuring the impact of small disjuncts on classifiers induced from thirty data sets.

5.2 Understanding Small Disjuncts

Danyluk and Provost (1993) observed that in the telecommunication domain they were studying, when they trained using noisy data, classifier accuracy suffered severely. They speculated that this occurred because: 1) it is difficult to distinguish between noise and true exceptions and, 2) in their domain, errors in measurement and classification often occur systematically rather than randomly. Thus, they speculated that it was difficult to distinguish between erroneous consistencies and correct ones.[2]

This speculation formed the basis for the research by Weiss (1995) and Weiss and Hirsh (1998). Weiss (1995) investigated the interaction between noise, rare cases and small disjuncts using synthetic datasets, for which the target concept is known and can be manipulated. Some synthetic data sets were generated based on concepts that included many rare cases (small subconcepts) while others were constructed from concepts that

[2] Throughout this thesis we use the term "small subconcepts" in place of the terms "rare cases", "true exceptions" and "correct exceptions" and the term "large subconcept" in place of "common cases" or "general cases". We believe this terminology is more precise and less prone to confusion.

included only general cases (large subconcepts). The research showed that the rare cases tended to form small disjuncts in the induced classifier. It further showed that systematic attribute noise, class noise and missing attributes can each cause small disjuncts to have higher error rates than large disjuncts and that those test examples associated with rare cases are misclassified more often than the test examples associated with common cases. Explanations for these observations were provided. For example, the impact of attribute noise was explained by noting that attribute noise can cause common cases in the training data to appear identical to rare cases, thus "overwhelming" the rare cases and causing the wrong class label to be assigned to the small disjuncts. This research was followed up by additional research that analyzed the impact of noise on learning and small disjuncts using two real-world data sets (Weiss & Hirsh, 1998). This research yielded conclusions similar to those found by Weiss (1995).

5.3 Addressing The Problem with Small Disjuncts

The majority of research on small disjuncts focuses on ways to address the problem with small disjuncts. Holte et al. (1989) evaluate several strategies for improving learning in the presence of small disjuncts. First, they show that the strategy of eliminating all small disjuncts is ineffective, because the emancipated examples are still likely to be misclassified. They then argue that while a maximum generality bias, which is used by systems such as ID3, is appropriate for large disjuncts, it is not appropriate for small disjuncts. Thus, they focus on a strategy for making small disjuncts highly specific. To test this claim, they run experiments where a maximum generality bias is used for the large disjuncts and a maximum specificity bias is used for the small disjuncts (for a maximum specificity bias all conditions satisfied by the training examples covered by a disjunct are

added to the disjunct). The experimental results show that with the maximum specificity bias, the resulting disjuncts cover fewer cases and have much lower error rates. Unfortunately, the emancipated examples increase the error rate of the large disjuncts, to the extent that the overall error rates remain roughly the same. Although the authors also experiment with a more selective bias that produces interesting results, it does not demonstrably improve classifier performance.

Ting (1994) evaluates a method for improving the performance of small disjuncts that also employs a maximum specificity bias. However, unlike the method employed by Holte et al., this method does not affect—and therefore cannot degrade—the performance of the large disjuncts. The basic approach is to first use C4.5 to determine if an example is covered by a small or large disjunct. If the example is covered by a large disjunct, then C4.5 is used to classify the example; otherwise an instance-based learner, IB1, is used to classify the example. Instance-based learning is used in this case because it is an extreme example of the maximum specificity bias.

In order to use Ting's hybrid learning method, there must be a specific criterion for deciding whether a disjunct is a small disjunct. Ting evaluated several criteria. These criteria used a threshold value, which was applied to 1) the absolute size of the disjunct, 2) the relative size of the disjunct, and 3) the error rate of the disjunct. For each criterion, only the best result, produced using the best threshold, was reported. Because the threshold values were selected using the test data rather than an independent hold-out set, the results are overly optimistic. Thus, although the observed results are encouraging, it cannot be claimed that the composite learner successfully addresses the problem with small disjuncts.

Carvalho and Freitas (2000) employ a hybrid method similar to that used by Ting. They also use C4.5 to build a decision tree and then, for each training example, use the size of the leaf covering that example to determine if the example is covered by a small or large disjunct. The training examples that fall into each small disjunct are then fed together into a genetic-algorithm based learner that forms rules to specifically cover the examples that fall into that individual disjunct. Test examples that fall into leaves corresponding to large disjuncts are then assigned a class label based on the decision tree; test examples that fall into a small disjunct are classified by the rules learned by the genetic algorithm for that particular disjunct. The experimental results are also encouraging, but because only a few data sets are analyzed, and because the improvements in classifier performance are only seen for certain specific definitions of "small disjunct", it cannot be concluded that this research substantially addresses the problem with small disjuncts.

Several other approaches have been proposed for addressing the problem with small disjuncts. Quinlan (1991) tries to minimize the problem by improving the probability estimates used to assign a class label to a disjunct. A naive estimate of the error rate of a disjunct is the proportion of the training examples that it misclassifies. However, this estimate performs quite poorly for small disjuncts, due to the small number of examples used to form the estimate. Quinlan describes a method for improving the accuracy estimates of the small disjuncts by taking the class distribution into account. The motivation for this work is that for unbalanced class distributions one would expect the disjuncts that predict the majority class to have a lower error rate than those predicting the minority class (this is the test distribution effect described in Chapter 6). Quinlan incorporates these prior probabilities into the error rate estimates. However, instead of using the over-

all class distribution as the prior probability, Quinlan generates a more representative measure by calculating the class distribution only on those training examples that are "close" to the small disjunct—that is, fail to satisfy at most one condition in the disjunct. The experimental results demonstrate that Quinlan's error rate estimation model outperforms the naive method, most significantly for skewed distributions. Note that this research describes a connection between the problem with small disjuncts and the class distribution of the data set. This connection was investigated further in Chapter 4, Section 4.

Van den Bosch, Weijters, Van den Herik and Daelemans (1997) advocate the use of instance-based learning for domains with many small disjuncts. They are mainly interested in language learning tasks, which they claim result in many small disjuncts, or “pockets of exceptions”. They focus on the problem of learning word pronunciations. Although instance-based learners may not appear to form disjunctive concepts, they do since they partition the instance space into disjoint regions. The authors compute cluster sizes, which they view as analogous to disjunct size. They determine cluster sizes by repeatedly selecting examples from the data and then form a ranked list of the 100 nearest neighbors. They then determine the rank of the nearest neighbor with a different class value—this value minus one is considered to be the cluster size. This method, as well as the more conventional method of measuring disjunct size via a decision tree, shows that the word pronunciation domain has many small disjuncts. The authors also try an information-theoretic weighted similarity matching function, which effectively re-scales the feature space so that "more important" features have greater weight. When this is done, the size of the average cluster increases from 15 to 25. Unfortunately, error rates were

not specified for the various clusters and hence one cannot measure how effective this strategy is for dealing with the problem with small disjuncts.

Chapter 6

The Role of Class Distribution in Learning

"…it's like finding a needle in a haystack."
- unknown

Class distribution plays a central role in learning. As we shall see, the class distribution of the training data affects a classifier's ability to classify minority-class examples and majority-class examples, as well as the overall performance of the induced classifier.

This chapter begins with a discussion of why training data is often costly and how this motivates the research in this thesis on class distribution. Basic terminology related to class distribution is then introduced. Next, the implications of altering the class distribution of the training data are discussed, as is a method for adjusting the induced classifier to account for these changes, so that the induced classifier is not improperly biased. The experimental methodology associated with the class distribution experiments is then described. The main experimental results for the chapter are then presented. These results analyze the differences in classifier performance, with respect to the minority and majority classes. The reasons for these differences are also discussed.

The analysis of class distribution and its effect on learning then continues in Chapter 7, which shows how varying the class distribution of the training data affects classifier performance. These results are used to identify and characterize the best class distribution for learning. Chapter 8 then provides a budget-sensitive progressive-sampling algo-

rithm for selecting examples, such that the resulting class distribution will yield classifiers with near optimal performance. Related research is described in Chapter 9.

It should be noted up front that this study of class distribution only utilizes C4.5, a decision-tree learner, and therefore, strictly speaking, the conclusions only apply to this class of learners. However, in Chapter 10 we describe why our results may hold for other learners as well.

6.1 Learning from Costly Data

As described in Chapter 1, there are two main motivations for studying the effect of class distribution on learning. The first reason concerns the need to provide a better understanding of class distribution and how it affects learning. The second reason has to do with selecting the class distribution carefully when the training data must be limited, so that classifier performance is not unnecessarily degraded. In this section we describe in much greater detail the reasons why the amount of training data may need to be limited—and how this relates to the study of class distribution.

In real world situations the amount of training data must often be limited because obtaining data in a form suitable for learning may be costly and/or learning from large amounts of data may be costly. The costs associated with forming a useful training set include the costs of obtaining the raw data, cleaning the data, transporting/storing the data, and transforming the data into a representation suitable for learning. The costs associated with learning from the data involve the cost of computer hardware, the "cost" associated with the time it takes to learn from the data, and the "opportunity cost" associated with suboptimal learning from extremely large data sets due to limited computational resources. Turney (2000) provides a more complete description of these costs.

When the costs of preparing or learning from the data are substantial, so that it is necessary to limit the amount of training data, an important question, investigated in this thesis is: in what proportion should the classes be represented in the training data? Some practitioners believe that the naturally occurring marginal class distribution should be used for learning, so that new examples will be classified using a model built from the same underlying distribution. Other practitioners believe that the training set should contain an increased percentage of minority-class examples. This viewpoint is supported by the following quote, "if the sample size is fixed, a balanced sample will usually produce more accurate predictions than an unbalanced 5%/95% split" (SAS, 2001). However, we are aware of no thorough empirical study of the relationship between the class distribution of the training set and classifier performance, so neither of these views has been validated and the choice of class distribution often is made arbitrarily—and with little understanding of the consequences. We remedy this in this thesis.

As just described, there are two basic scenarios where the amount of training data may need to be limited. The first scenario is when the cost of learning from the data (e.g., computation costs) is costly. In this case the research presented in this thesis will help determine which existing training examples to *discard*, so that the resulting class distribution yield a classifier that performs well. The second scenario is when it is costly to obtain usable training examples. In this situation the research in this thesis will help determine the proportion of examples belonging to each class to *procure*, again so that the induced classifier will perform well. However, this assumes that one can select examples belonging to a specific class—and only incur the acquisitions costs for examples belonging to this class. There are many real world situations where this is the case. This occurs

when the examples belonging to each class come from separate sources or are generated by separate processes, or when the cost of procuring the example is due to the cost of forming a useful training example rather than the cost of obtaining the raw, labeled, data.

Telecommunication fraud detection (Fawcett & Provost, 1997) provides an example where data can be obtained from separate sources. In this domain, any transactions stated to be fraudulent by the customer are first verified to ensure they are not legitimate and then are stored separately from the valid transactions. Because of this, fraudulent transactions can be procured independently of non-fraudulent transactions.

In other situations labeled raw data can be obtained very cheaply, but it is the process of forming usable training examples from the raw data that is expensive. For example, consider the *phone* data set, one of the twenty-six data sets used in this thesis to study class distribution. This data set is constructed from low-level call-detail records that describe a phone call, where each call-detail record includes the originating and terminating phone numbers, the connection time, the day of week and duration of each call. The learning task is to determine whether a phone line, based on its pattern of usage, belongs to a business or a residence. Each phone line may have hundreds or even thousands of call-detail records associated with it, and all of these must be aggregated/summarized to form a single training example. Billions of call-detail records, covering hundreds of millions of phone lines, potentially are available for learning. Because of the effort associated with loading data from dozens of computer tapes, disk-space limitations, and the enormous processing time required to summarize the raw data, it is not feasible to construct a data set using all available raw data. Consequently, the number of training examples must be limited. Since the class label associated with each phone line is known

for the training data, it is straightforward to select training examples (i.e., phone lines) based on the class. Because of the costs just described, the phone data set was limited to approximately 650,000 training examples, which were generated from approximately 600 million call-detail records (a small fraction of the call-detail records available). Given the number of large transaction-oriented databases in existence, this need to limit the number of usable (i.e., summarized) training examples surely is not uncommon.

6.2 A Quick Example—Why Does Class Distribution Matter?

To provide some initial insight to why one might want to choose the class distribution of the training data carefully, given a fixed amount of training data, consider Figure 6.1. In addition to displaying a traditional learning curve, which shows how the overall accuracy of the induced classifier varies in response to changes in training-set size, this figure also includes learning curves that show the ability of the classifier to correctly classify minority-class and majority-class test examples.

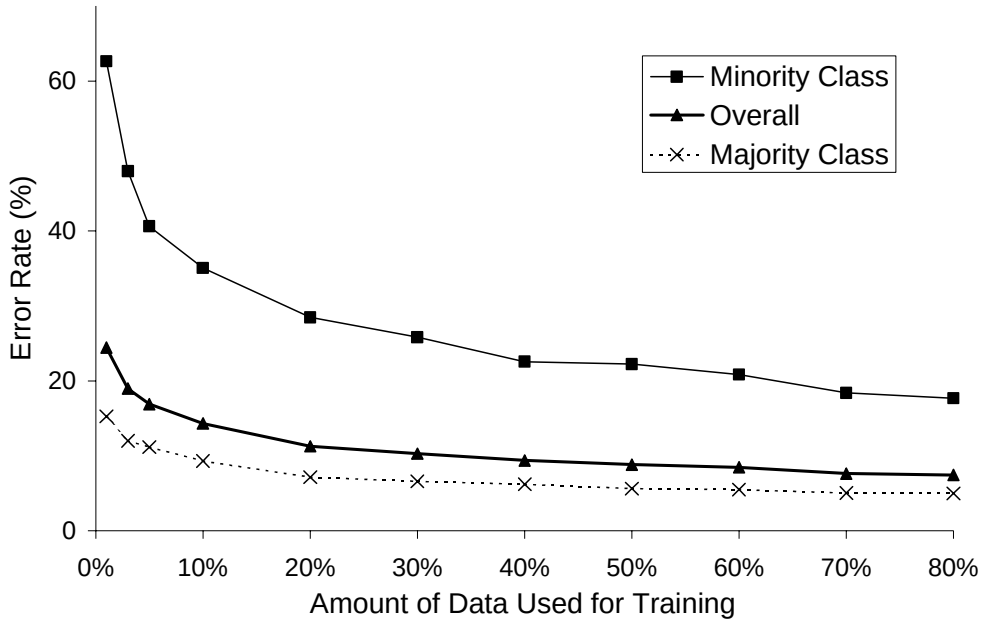

Figure 6.1: Learning Curves for the Letter-Vowel Data Set

Figure 6.1 demonstrates that providing additional training examples affects the performance of the minority and majority classes differently (i.e., the slopes of the corresponding learning curves are different). While this does not necessarily tell us from which class to choose additional examples (because the two classes are not learned independently), it does indicate that adding examples from each class in a proportion that is *different* from the naturally occurring class distribution will lead to classifiers with different performance characteristics. It seems reasonable to expect that *some* class distribution may yield a classifier that outperforms the one induced from the naturally occurring class distribution.

6.3 Terminology

This section introduces some additional terminology for two class learning problems, especially as it relates to class distribution. This terminology will help with the analysis of how class distribution affects learning. In the following, recall that the positive class corresponds to the minority class and that the negative class corresponds to the majority class.

Formally, let x be an instance drawn from some fixed distribution D. Every instance x is mapped (perhaps probabilistically) to a class $C \in \{p, n\}$ by a function c, where c represents the true, but unknown, classification function. Let ρ be the marginal probability of membership of x in the positive class. The marginal probability of membership in the negative class equals $1 - \rho$. These marginal probabilities sometimes are referred to as the "class priors" or the "base rate".

A classifier t is a mapping from instances x to classes $\{p, n\}$ and is an approximation of c. For notational convenience, let $t(x) \in \{P, N\}$ so that it is always clear whether a

class value is an actual (lower case) or predicted (upper case) value. The expected accuracy of a classifier t, α_t, is defined as $\alpha_t = Pr(t(x) = c(x))$, or, equivalently as:

$$\alpha_t = \rho \bullet Pr(t(x) = P \mid c(x) = p) + (1 - \rho) \bullet Pr(t(x) = N \mid c(x) = n) \quad [6.1]$$

Many classifiers produce not only a classification, but also estimates of the probability that x will take on each class value. Let $Post_t(x)$ be classifier t's estimated (posterior) probability that for instance x, $c(x)=p$. Classifiers that produce class-membership probabilities produce a classification by applying a numeric threshold to the posterior probabilities. For example, a threshold value of .5 may be used so that $t(x) = P$ *iff* $Post_t(x) > .5$; otherwise $t(x) = N$.

A variety of classifiers function by partitioning the input space into a set $\mathcal{L}$ of disjoint regions (a region being defined by a set of potential instances). For example, for a classification tree, the regions are described by conjoining the conditions leading to the leaves of the tree. Each region $L \in \mathcal{L}$ will contain some number of training instances, λ_L. Let λ_{Lp} and λ_{Ln} be the numbers of positive and negative training instances in region L, such that $\lambda_L = \lambda_{Lp} + \lambda_{Ln}$. Such classifiers often estimate $Post_t(x \mid x \in L)$ as $\lambda_{Lp}/(\lambda_{Lp} + \lambda_{Ln})$ and assign a classification for all instances $x \in L$ based on this estimate and a numeric threshold, as described earlier. Now, let $\mathcal{L}_P$ and $\mathcal{L}_N$ be the sets of regions that predict the positive and negative classes, respectively, such that $\mathcal{L}_P \cup \mathcal{L}_N = \mathcal{L}$. For each region $L \in \mathcal{L}$, t has an associated accuracy, $\alpha_L = Pr(c(x) = t(x) \mid x \in L)$. Let $\alpha_{\mathcal{L}_P}$ represent the expected accuracy for $x \in \mathcal{L}_P$ and $\alpha_{\mathcal{L}_N}$ the expected accuracy for $x \in \mathcal{L}_N$ (for notational conven-

ience we treat $\mathcal{L}_P$ and $\mathcal{L}_N$ as the union of the set of instances in the corresponding regions).

6.4 Correcting for Changes to the Class Distribution

Many classifier induction algorithms assume that the training and test data are drawn from the same fixed, underlying, distribution *D.* In particular, these algorithms assume that r_{train} and r_{test}, the fractions of positive examples in the training and test sets, approximate ρ, the true "prior" probability of encountering a positive example. These induction algorithms use the estimated class priors based on r_{train}, either implicitly or explicitly, to construct a model and to assign classifications. If the estimated value of the class priors is not accurate, then the posterior probabilities of the model will be improperly biased. Specifically, "increasing the prior probability of a class increases the posterior probability of the class, moving the classification boundary for that class so that more cases are classified into the class" (SAS, 2001). Thus, if the training-set data are selected so that r_{train} does not approximate ρ, then the posterior probabilities should be adjusted based on the differences between ρ and r_{train}. If such a correction is not performed, then the resulting bias will cause the classifier to classify the preferentially sampled class more accurately, but the overall accuracy of the classifier will almost always suffer (we show this later in this chapter). Thus, it is critical that the classifier be adjusted to eliminate this bias.[3]

[3] In situations where it is more costly to misclassify minority-class examples than majority-class examples, practitioners sometimes introduce this bias on purpose. We would argue that in this case a more appropriate approach would be to use a probabilistic or cost-sensitive learning method, so that all available training data can be used for training.

In the remainder of this section we describe a method for adjusting the posterior probabilities to account for the difference between r_{train} and ρ so that this bias is avoided. For concreteness, this method is described with respect to a decision tree classifier, but the description applies equally well to any classifier that operates by partitioning the input space into regions. This method, previously documented (Weiss & Provost, 2001), is justified informally, using a simple, intuitive, argument. Elkan (2001) presents an equivalent method for adjusting the posterior probabilities and includes a formal derivation.

Differences between r_{train} and ρ will normally result in biased posterior class-probability estimates at the leaves of a decision tree. To remove this bias, the probability estimates are adjusted to take these differences into account. Two common probability estimation formulas are described in Table 6.1. For each, let λ_{Lp} (λ_{Ln}) represent the number of minority-class (majority-class) training examples at a leaf L of a decision tree (or, more generally, within any partition L). The uncorrected estimates, which are in common use, estimate the probability of seeing a minority-class (positive) example in L. These uncorrected estimates are based on the assumption that the training and test sets are drawn from the same population and therefore both approximate ρ. The uncorrected frequency-based estimate is straightforward and requires no explanation. However, this estimate does not perform well when the sample size, $\lambda_{Lp}+\lambda_{Ln}$, is small—and is not even defined when the sample size is 0. For these reasons the Laplace estimate often is used instead. We consider a version based on the Laplace law of succession (Good, 1965). This probability estimate will always be closer to 0.5 than the frequency-based estimate, but the difference between the two estimates will be negligible for large sample sizes.

Estimate Name	**Uncorrected**	**Corrected**
Frequency-Based	$\lambda_{Lp}/(\lambda_{Lp}+\lambda_{Ln})$	$\lambda_{Lp}/(\lambda_{Lp}+o\ \lambda_{Ln})$
Laplace (law of succession)	$(\lambda_{Lp}+1)/(\lambda_{Lp}+\ \lambda_{Ln}+\ 2)$	$(\lambda_{Lp}+1)/(\lambda_{Lp}+o\ \lambda_{Ln}+\ 2)$

Table 6.1: Probability Estimates for Observing a Minority-Class Example

The corrected versions of the estimates in Table 6.1 account for differences between r_{train} and ρ by factoring in the over-sampling ratio *o*, which measures the degree to which the minority class is over-sampled in the training set relative to the naturally occurring distribution. The value of *o* is computed as the ratio of minority-class examples to majority-class examples in the training set divided by the same ratio in the naturally occurring class distribution. If the ratio of minority to majority examples were 1:2 in the training set and 1:6 in the naturally occurring distribution, then *o* would be 3. A learner can account properly for differences between r_{train} and ρ by using the corrected estimates to calculate the posterior probabilities at L.

As an example, if the ratio of minority-class examples to majority-class examples in the naturally occurring class distribution is 1:5 but the training distribution is modified so that the ratio is 1:1, then *o* is 1.0/0.2, or 5. For L to be labeled with the minority class the probability must be greater than 0.5, so, using the corrected frequency-based estimate, $\lambda_{Lp}/(\lambda_{Lp}+5\lambda_{Ln}) > 0.5$, or, $\lambda_{Lp} > 5\ \lambda_{Ln}$. Thus, L is labeled with the minority class only if it covers *o* times as many minority-class examples as majority-class examples. This result should make intuitive sense.

Note that in calculating *o* class ratios are used instead of the fraction of examples belonging to the minority class (if we mistakenly used the latter in the above example, then

o would be one-half divided by one-sixth, or 3). Using the class ratios substantially simplifies the formulas and leads to more easily understood estimates. Elkan (2001) provides a more complicated, but equivalent, formula that uses fractions instead of ratios.

In this discussion we assume that a good approximation of the true base rate is known. In some real-world situations this is not true and different methods are required to compensate for changes to the training set (Provost et al., 1998; Saerens et al., 2002).

6.5 Experimental Methodology

This section describes the experimental methodology associated with the class distribution experiments. First, the methodology for altering the class distribution of the training set is described. Then the changes to C4.5 to account for changes to the class distribution of the training set are summarized. Finally, the use of pruning is described.

6.5.1 Methodology for Altering the Class Distribution of the Training Data

In order to investigate the effect that different training-set class distributions have on classifier performance, it is necessary to have a method for generating training sets with a variety of class distributions. So that the classifiers induced from these different distributions can be compared fairly, it is essential that all training sets associated with each data set contain the same number of examples. Furthermore, examples should not be duplicated in order to achieve the desired class distribution—this distorts the learning problem and makes it appear easier than it actually is. For example, if a single example is copied twenty times, the learner might quite possibly form a disjunct to cover that one "distinct" example. This overfitting would lead to good classification performance on the training data, but this would not carry over to new, unseen, examples.

The following procedure for creating the training and test sets will satisfy all of the requirements just described. First, the test set is formed by randomly selecting 25% of the minority-class examples and 25% of the majority-class examples from the original data set, without replacement. The resulting test set therefore conforms to the original class distribution. The remaining data are then available for training. To ensure that all experiments for a given data set have the same training-set size—no matter what the class distribution of the training set—the size of the training set, S, is made equal to the total number of minority-class examples still available for training (i.e., 75% of the original number). This makes it possible, without replicating any examples, to generate *any* class distribution for training-set size S (i.e., from 100% minority-class examples to 100% majority-class examples). The training set is then formed by stratified random sampling from the remaining data, without replacement, such that the desired class distribution is achieved.

6.5.2 Classifier Modifications to Account for Changes in Class Distribution

C4.5 always assumes that the training data approximate the true, underlying distribution, and hence uses the uncorrected frequency-based estimate to label the leaves of the decision tree. When the class distribution of the training set altered, it is essential that the corrected version of the estimate be used to label the leaves of the induced decision tree.

The C4.5 source code is not modified to use the corrected estimates to label the leaves of the classifier; instead, once C4.5 completes, a post-processing step is executed that relabels the leaves of the decision tree using the corrected estimate. This post-processing step then recalculates all classifier performance statistics and records the percentage of leaves that are assigned a different class label due to the use of the corrected estimate.

The change in error rate that results from the relabeling process is also recorded. This information is used in Section 6.6.3 to show that the use of the corrected probability estimates consistently yields a substantial improvement in classifier performance.

The results presented in the remainder of the thesis, unless otherwise noted, are based on the use of the corrected versions of the frequency-based and Laplace estimates with a probability threshold of .5 to label the leaves of the induced decision trees. The frequency-based version is employed when accuracy is used to measure classifier performance. The Laplace version is employed for generating the ROC curves when AUC is used to measure classifier performance, since in this situation the Laplace estimate has been shown to yield consistent improvements in performance (Provost & Domingos, 2001).

6.5.3 Issues with Pruning

If C4.5's pruning strategy, which attempts to minimize error rate, were allowed to execute, it would prune based on a false assumption (viz., that the test distribution matches the training distribution). Since this may negatively affect the generated classifier, except where otherwise indicated, all results are based on C4.5 without pruning. This decision is supported by recent research, which indicates that when target misclassification costs (or class distributions) are unknown then standard pruning should be avoided (Provost & Domingos, 2001; Zadrozny & Elkan, 2001; Bradford et al, 1998; Bauer & Kohavi, 1999). Indeed, Bradford et al. found that even if the pruning strategy is adapted to take misclassification costs and class distribution into account, this does not generally improve the performance of the classifier. Nonetheless, in order to justify the validity of the results when using C4.5 without pruning, we also present the results of C4.5 with pruning, when

the training set uses the natural distribution. In this situation C4.5's assumption about r_{train} approximating ρ is valid and hence C4.5's pruning strategy will function appropriately. Looking forward, these results show that in this situation C4.5 without pruning performs competitively with C4.5 with pruning. Based on this we conclude that C4.5 without pruning is a reasonable learner.

6.6 Measuring the Impact of Class Distribution on Classifier Performance

We now are ready to analyze the classifiers induced from the twenty-six naturally unbalanced data sets described earlier in Table 2.4. In this section the focus is on identifying and explaining any differences in classification performance between the minority and majority classes. However, before addressing these differences, it is important to discuss an issue that may lead to confusion if left untreated. Practitioners have noted that learning performance often is unsatisfactory when learning from data sets where the minority class is considerably underrepresented. In particular, they observe that there is a large error rate for the minority class. As should be clear from the evaluation metrics described in Table 2.2 and the associated discussion, there are two different notions of "error rate for the minority class": the minority-class predictions could have a high error rate (large $\overline{PPV}$) or the minority-class test examples could have a high error rate (large FN). When practitioners observe that the error rate is unsatisfactory for the minority class, they are usually referring to the fact that the minority-class *examples* have a high error rate (large FN). The analysis in this section will show that the error rate associated with the minority-class predictions *and* the minority-class test examples are both much larger than their majority class counterparts. Explanations for these observed differences are provided.

6.6.1 Experimental Results

The performance of the classifiers induced from the twenty-six, naturally unbalanced, data sets is described in Table 6.2.

Dataset	% Minority Examples	% Errors from Min.	Leaves		Coverage		Prediction ER		Actuals ER	
			Min.	Maj.	Min.	Maj.	Min.	Maj.	Min.	Maj.
							$\overline{(PPV)}$	$\overline{(NPV)}$	(FN)	(FP)
letter-a	3.9	58.3	11	138	2.2	4.3	32.5	1.7	41.5	1.2
pendigits	8.3	32.4	6	8	16.8	109.3	25.8	1.3	14.3	2.7
abalone	8.7	68.9	5	8	2.8	35.5	69.8	7.7	84.4	3.6
sick-euthyroid	9.3	51.2	4	9	7.1	26.9	22.5	2.5	24.7	2.4
connect-4	9.5	51.4	47	128	1.7	5.8	55.8	6.0	57.6	5.7
optdigits	9.9	73.0	15	173	2.9	2.4	18.0	3.9	36.7	1.5
covertype	14.8	16.7	350	446	27.3	123.2	23.1	1.0	5.7	4.9
solar-flare	15.7	64.4	12	48	1.7	3.1	67.8	13.7	78.9	8.1
phone	18.2	64.4	1008	1220	13.0	62.7	30.8	9.5	44.6	5.5
letter-vowel	19.4	61.8	233	2547	2.4	0.9	27.0	8.7	37.5	5.6
contraceptive	22.6	48.7	31	70	1.8	2.8	69.8	20.1	68.3	21.1
adult	23.9	57.5	627	4118	3.1	1.6	34.3	12.6	41.5	9.6
splice-junction	24.1	58.9	26	46	5.5	9.6	15.1	6.3	20.3	4.5
network2	27.9	57.1	50	61	4.0	10.3	48.2	20.4	55.5	16.2
yeast	28.9	58.9	8	12	14.4	26.1	45.6	20.9	55.0	15.6
network1	29.2	57.1	42	49	5.1	12.8	46.2	21.0	53.9	16.7
car	30.0	58.6	38	42	3.1	6.6	14.0	7.7	18.6	5.6
german	30.0	55.4	34	81	2.0	2.0	57.1	25.4	62.4	21.5
breast-wisc	34.5	45.7	5	5	12.6	26.0	11.4	5.1	9.8	6.1
blackjack	35.6	81.5	13	19	57.7	188.0	28.9	27.9	64.4	8.1
weather	40.1	50.7	134	142	5.0	7.2	41.0	27.7	41.7	27.1
bands	42.2	91.2	52	389	1.4	0.3	17.8	34.8	69.8	4.9
market1	43.0	50.3	87	227	5.1	2.7	30.9	23.4	31.2	23.3
crx	44.5	51.0	28	65	3.9	2.1	23.2	18.9	24.1	18.5
kr-vs-kp	47.8	54.0	23	15	24.0	41.2	1.2	1.3	1.4	1.1
move	49.9	61.4	235	1025	2.4	0.6	24.4	29.9	33.9	21.2
Average	25.8	56.9	120	426	8.8	27.4	33.9	13.8	41.4	10.1
Median	26.0	57.3	33	67	3.9	6.9	29.9	11.1	41.5	5.9

Table 6.2: Behavior of Classifiers Induced from Unbalanced Data Sets

This table warrants some explanation. The first column specifies the data set name while the second column specifies the percentage of minority-class examples in natural class distribution. The third column specifies the percentage of the total test errors that can be attributed to misclassified minority-class test examples. By comparing the values in col-

umns two and three we see that in all cases a disproportionately large percentage of the errors come from the minority-class examples. For example, minority-class examples make up only 3.9% of the letter-a data set but contribute 58.3% of the errors. Furthermore, for 22 of 26 data sets a *majority* of the errors can be attributed to minority-class examples.

The fourth column in Table 6.2 specifies the number of leaves labeled with the minority and majority classes and shows that in all but two cases there are fewer leaves labeled with the minority class than with the majority class. The fifth column, "Coverage," specifies the average number of training examples that each minority-labeled or majority-labeled leaf classifies ("covers"). These results indicate that the leaves labeled with the minority class are formed from far fewer training examples than those labeled with the majority class.

The "Prediction ER" column specifies the error rates associated with the minority-class and majority-class predictions, based on the performance of these predictions at classifying the test examples. The "Actuals ER" column specifies the classification error rates for the minority and majority class examples, again based on the test set. These last two columns are also labeled using the terms defined in Table 2.2 ($\overline{\text{PPV}}$, $\overline{\text{NPV}}$, FN, and FP). As an example, these columns show that for the letter-a data set the minority-labeled predictions have an error rate of 32.5% while the majority-labeled predictions have an error rate of only 1.7%, and that the minority-class test examples have a classification error rate of 41.5% while the majority-class test examples have an error rate of only 1.2%. In each of the last two columns we underline the higher error rate.

The results in Table 6.2 clearly demonstrate that the minority-class predictions perform much worse than the majority-class predictions and that the minority-class examples are misclassified much more frequently than majority-class examples. Over the twenty-six data sets, the minority predictions have an average error rate ($\overline{\text{PPV}}$) of 33.9% while the majority-class predictions have an average error rate ($\overline{\text{NPV}}$) of only 13.8%. Furthermore, for only three of the twenty-six data sets do the majority-class predictions have a higher error rate—and for these three data sets the class distributions are only slightly unbalanced. Table 6.2 also shows us that the average error rate for the minority-class test examples (FN) is 41.4% whereas for the majority-class test examples the error rate (FP) is only 10.1%. In every one of the twenty-six cases the minority-class test examples have a higher error rate than the majority-class test examples.

6.6.2 Discussion of Results

So why do the minority-class predictions have a higher error rate ($\overline{\text{PPV}}$) than the majority-class predictions ($\overline{\text{NPV}}$)? There are at least two reasons. First, consider a classifier t_{random} where the partitions $\mathcal{L}$ are chosen randomly and the assignment of each $L \in \mathcal{L}$ to $\mathcal{L}_P$ and $\mathcal{L}_N$ is also made randomly (recall that $\mathcal{L}_P$ and $\mathcal{L}_N$ represent the regions labeled with the positive and negative classes). For a two-class learning problem the expected overall accuracy, α_t, of this *randomly* generated and labeled classifier must be 0.5. However, the expected accuracy of the regions in the positive partition, $\alpha_{\mathcal{L}_P}$, will be ρ while the expected accuracy of the regions in the negative partition, $\alpha_{\mathcal{L}_N}$, will be $1 - \rho$. For a highly unbalanced class distribution where ρ=.01, $\alpha_{\mathcal{L}_P} = .01$ and $\alpha_{\mathcal{L}_N} = .99$. Thus, in such a scenario the negative/majority predictions will be much more "accurate." While this

"test distribution effect" will be small for a well-learned concept with a low Bayes error rate (and non-existent for a perfectly learned concept with a Bayes error rate of 0), many learning problems are quite hard and have high Bayes error rates.[4]

The results in Table 6.2 suggest a second explanation for why the minority-class predictions are so error prone. According to the coverage results, minority-labeled predictions tend to be formed from fewer training examples than majority-labeled predictions. Hence minority-labeled predictions are more often associated with small disjuncts than majority-labeled predictions. Since small disjuncts, as shown in Chapter 3, have a much higher error rate than large disjuncts, rules/leaves labeled with the minority class have a higher error rate in part because they suffer more from this "problem of small disjuncts." This helps to explain the results pertaining to the effect of class imbalance on small disjuncts, summarized earlier in Chapter 4, Figure 4.13. These results showed that naturally unbalanced data sets yield classifiers with higher error concentrations than the balanced versions of the same data sets. The explanation for this behavior is that because the test distribution effect makes the minority-class examples harder to learn, it also makes small disjuncts harder to learn for the unbalanced data sets—because the small disjuncts are disproportionately labeled with the minority class.

Next, we consider why minority-class test examples are misclassified much more often than majority-class test examples (FN > FP)—a phenomenon that has also been observed by others (Japkowicz & Stephen, 2002). We begin by re-expressing the estimated

[4] The (optimal) Bayes error rate, using the terminology from Chapter 2, occurs when t(.)=c(.). Because c(.) may be probabilistic (e.g., when noise is present), the Bayes error rate for a well-learned concept may not always be low. The test distribution effect will only be small when the concept is well learned *and* the Bayes error rate is low.

accuracy, a_t, of a classifier *t*, in terms of r_{test} (the fraction of positive examples in the test set), where the test set is drawn from the true, underlying distribution *D*:

$$a_t = TP \bullet r_{test} + TN \bullet (1 - r_{test}) \quad [6.2]$$

Since the positive class corresponds to the minority class, $r_{test} < .5$ and for highly unbalanced data sets $r_{test} << .5$. Thus, TN, the true negative rate is weighted more than TP, the true positive rate. Given the definitions of these terms in Table 2.2, this means that a strategy to maximize accuracy will place greater emphasis on maximizing the number of true negatives (tn) than maximizing the number of true positives (tp), and also more emphasis on minimizing the number of false positives (fp) than on minimizing the number of false negatives (fn). An induction algorithm geared toward maximizing accuracy therefore should "prefer" false-negative errors over false-positive errors. This will cause negative/majority examples to be predicted more often and hence will lead to a higher error rate for minority-class examples. One straightforward example of how learning algorithms exhibit this behavior is provided by the common-sense rule: if there is no evidence favoring one classification over another, then predict the majority class. This also explains why, when learning from data sets with a high degree of class imbalance, classifiers rarely predict the minority class.

A second reason why minority-class examples are misclassified more often than majority-class examples is that fewer minority-class examples are likely to be sampled from the distribution *D*. Therefore, the training data are less likely to include (enough) instances of all of the minority-class subconcepts in the concept space, and the learner may not have the opportunity to represent all truly positive regions in $\mathcal{L}_P$. Because of this,

some minority-class test examples will be mistakenly classified as belonging to the majority class.

It is worth noting that $\overline{\text{PPV}} > \overline{\text{NPV}}$ does not imply that FN > FP. That is, having more error-prone minority predictions does *not* imply that the minority-class examples will be misclassified more often than majority-class examples. Indeed, a higher error rate for minority predictions means more majority-class test examples will be misclassified. The reason we generally observe a lower error rate for the majority-class test examples (FN > FP) is because the majority class is predicted far more often than the minority class.

Finally, these experiments were repeated with pruning. The results showed the same general trends but the magnitudes of the differences were altered. With pruning the error rate associated with the minority-class predictions ($\overline{\text{PPV}}$) decreases dramatically while the average error rate of the majority-class predictions ($\overline{\text{NPV}}$) decreases only slightly, so that even though $\overline{\text{PPV}} > \overline{\text{NPV}}$, the magnitude of the difference is, on average, cut in half. Pruning also causes the error rate for the minority-class test examples (FN) to increase and the error rate for the majority-class test examples (FP) to decrease, so that the average difference becomes slightly larger. Both of these differences can be explained by the effect that pruning has on small disjuncts. According to the results in Chapter 4, pruning eliminates many, if not most, small disjuncts and many of the emancipated examples are then classified by the larger disjuncts. Because minority-class leaves tend to have smaller-sized disjuncts, pruning tends to eliminate minority-labeled leaves disproportionately. This further increases the error rate for the minority-class test examples and decreases the error rate of the majority-class test examples (since an increased number of examples will be classified as belonging to the majority class). The error rate of the mi-

nority-class predictions is reduced because the most error-prone of these predictions, which tend to be associated with the smallest disjuncts, are eliminated.

6.6.3 The Impact of the Class Distribution Correction on Classifier Performance

The experimental results presented in Section 6.6.1 are based on the use of the corrected frequency-based estimate to label the leaves of the induced decision trees. This corrected estimate ensures that the decision trees are not improperly biased by the changes made to the class distribution of the training set. Table 6.3 compares the performance of the decision trees labeled using the uncorrected frequency-based estimate (FB) with those labeled using the corrected frequency-based method (CT-FB). This comparison demonstrates the impact that the corrected estimate has on classifier performance. The comparison is based on the scenario where the training set is altered so that it uses a balanced class distribution rather than the naturally occurring class distribution. The results are based on 30 runs and the data sets are listed in order of decreasing class imbalance.

The error rates for the uncorrected and corrected frequency based estimates are displayed in the second and third columns of Table 6.3, respectively, and for each data set the lowest error rate is underlined. The fourth column specifies the relative improvement that results from using the corrected frequency-based estimate. The fifth column specifies the percentage of the leaves in the decision tree that are assigned a different class label when the corrected estimate is used. The last two columns specify, for each estimate, the percentage of the total errors that are contributed by the minority-class test examples.

Dataset	Error Rate		% Rel.	% Labels	% Errors from Min.	
	FB	CT-FB	Improv.	Changed	FB	CT-FB
letter-a	9.79	5.38	45.0	39.0	2.7	7.2
pendigits	4.09	4.02	1.7	3.2	5.6	7.8
abalone	30.45	22.97	24.6	5.6	8.5	19.1
sick-euthyroid	9.82	6.85	30.2	6.7	8.8	14.6
connect-4	30.21	27.57	8.7	14.7	8.5	10.4
optdigits	6.17	3.41	44.7	42.5	6.0	21.2
covertype	6.62	6.46	2.4	2.4	7.0	8.5
solar-flare	36.20	29.12	19.6	20.4	19.3	30.7
phone	17.85	14.81	17.0	3.2	25.2	44.4
letter-vowel	18.89	14.16	25.0	44.1	15.9	30.2
contraceptive	40.77	39.65	2.7	11.1	20.6	27.6
adult	22.69	20.05	11.6	30.7	19.6	36.8
splice-junction	9.02	8.74	3.1	14.1	20.1	28.4
network2	30.80	29.96	2.7	1.2	32.9	40.1
yeast	34.01	28.80	15.3	4.6	29.4	47.0
network1	31.99	30.99	3.1	1.3	32.9	38.2
car	8.26	7.92	4.1	5.3	25.9	33.8
german	38.37	37.09	3.3	16.1	30.8	35.8
breast-wisc	6.76	6.74	0.3	0.4	38.5	38.7
blackjack	33.02	28.71	13.1	17.1	42.9	76.2
weather	34.62	34.61	0.0	0.0	40.5	40.5
bands	32.68	32.68	0.0	0.6	90.2	90.2
market1	25.77	25.77	0.0	23.9	46.0	48.6
crx	20.84	21.48	-3.1	17.2	46.2	51.4
kr-vs-kp	1.22	1.22	0.0	0.2	58.5	58.5
move	28.24	28.24	0.0	20.8	52.6	60.7
Average	21.89	19.90	10.6	13.3	28.3	36.4

Table 6.3: Impact of the Probability Estimates on Error Rate

Table 6.3 shows that by employing the corrected frequency-based estimate instead of the uncorrected frequency-based estimate, there is, on average, a relative 10.6% reduction in error rate. Furthermore, in only one case does the uncorrected frequency-based estimate outperform the corrected frequency-based estimate. The correction tends to yield a larger reduction for the most highly unbalanced data sets—in which cases it plays a larger role. If we restrict ourselves to the first 13 data sets listed in Table 2.4, for which the minority-class makes up less than 25% of the examples, then the relative improvement over these data sets is 18.2%. Note that because in this scenario the minority class is oversampled in the training set, the corrected frequency-based estimate can only cause a label

to change from the minority class to the majority class. Consequently, as the last column in the table demonstrates, the corrected version of the estimate will cause more of the errors to come from the minority-class test examples. This generally yields an improvement in accuracy because accuracy places greater weight on the ability to classify the more numerous majority-class examples.

These results demonstrate that it is critical to account for changes made to the class distribution of the training data. Previous work on modifying the class distribution of the training set (Catlett, 1991; Chan & Stolfo, 1998; Japkowicz, 2002) did not take these differences into account and this undoubtedly affected the results.

6.7 Summary

This chapter covered a great deal of material. It began with a discussion of why training data may be costly and showed how this motivates the research in this thesis on class distribution. Namely, we saw that when data is costly, choosing the class distribution carefully may allow one to compensate for a reduced amount of training data (Chapter 7 will show that this is actually achievable). The implications of modifying the class distribution of the training set were discussed, as was a method for accounting for this change in distribution. This method involved re-calculating the class-probability estimates generated by the classifier, using probability estimates that are sensitive to the changes made to the class distribution of the training data.

The main experimental results for this chapter were then presented. These results described the performance of classifiers induced from twenty-six data sets, with respect to differences in the minority and majority classes. The results showed that minority-class predictions consistently have a much higher error rate than majority-class predictions,

and that minority-class test examples are misclassified much more often than majority-class test examples. A detailed discussion of these results followed, along with an explanation for the observed differences between the two classes. Finally, the performance of the classifiers labeled using the corrected probability estimate was compared to the performance of those classifiers labeled without the correction. These results clearly showed it is essential to account for changes made to the class distribution of the training data—not doing so consistently degrades classifier performance.

Chapter 7

The Effect of Class Distribution on Classifier Performance

If you choose not to decide—you still have made a choice!

- Neal Peart

In this chapter we turn to the central questions regarding class distribution: how does the class distribution of the training data affect classifier performance and what class distribution produces the best classifier? This chapter begins by describing how to determine which class distribution(s) yield the best classifier. This is followed by the main experimental results for the chapter, which measures classifier performance for twenty-six data sets using a variety of different training-set class distributions. These results show that the best class distribution for learning generally is not the naturally occurring class distribution. Finally, the relationship between training-set size, class distribution, and classifier performance is empirically investigated for three data sets. These results show that the best class distribution for learning does not vary dramatically as the amount of training data grows and that improved classifier performance is possible with less training data if the class distribution is chosen carefully.

7.1 Methodology for Determining the Optimum Class Distribution

In order to evaluate the effect of class distribution on classifier performance, the class distribution of the training data is varied for the twenty-six data sets using the methodology described in Chapter 6. The following twelve class distributions, expressed as the percentage of minority-class examples in the training data, are evaluated: 2%, 5%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, and 95%. The naturally occurring class distribution for each data set is also evaluated.

Before one can determine the "best" class distribution for learning, there are several issues that must be addressed. First, because we do not evaluate every possible class distribution, we can only determine the best distribution among the 13 evaluated distributions. Beyond this concern, however, is the issue of statistical significance and, because we generate classifiers for 13 training distributions, the issue of multiple comparisons (Jensen & Cohen, 2000). Because of these issues we cannot always conclude that the distribution that yields the best performing classifiers is truly the best one for training.

Several steps are taken to address the issues of statistical significance and multiple comparisons. To enhance our ability to identify true differences in classifier performance with respect to changes in class distribution, all results presented in this chapter are based on 30 runs, rather than the 10 runs employed in Chapter 6. Also, rather than trying to determine *the* best class distribution, we take a more conservative approach, and instead identify an "optimal range" of class distributions—a range in which we are confident the best distribution lies.

In order to identify the optimal range of class distributions, we begin by identifying, for each data set, the class distribution that yields the classifiers that perform best over the

30 runs. Then t-tests are performed to compare the performance of these 30 classifiers with the 30 classifiers generated using each of the other twelve class distributions (i.e., 12 t-tests each with n=30 data points). If a t-test yields a probability ≤ .10 then we conclude that the "best" distribution is different from the "other" distribution (i.e., we are at least 90% confident of this); otherwise we cannot conclude that the class distributions truly perform differently and therefore "group" the distributions together. These grouped distributions collectively form the "optimal range" of class distributions. As will be seen in the two tables that follow, in 50 of 52 cases the optimal ranges are contiguous, assuaging concerns that our conclusions are due to problems with multiple comparisons.

7.2 The Effect of Class Distribution on Error Rate

Table 7.1 displays the error rates of the classifiers induced for each of the twenty-six data sets using the thirteen class distributions. The first column in Table 7.1 specifies the data set name and the next two columns specify the error rates that result from using the natural distribution, with and then without pruning. The next 12 columns present the error rate values for the 12 fixed class distributions (without pruning). The "best" class distribution for each data set (i.e., the one that yields the lowest error rate) is highlighted by underlining it and displaying it in boldface. The relative position of the natural distribution within the range of evaluated class distributions is denoted by the use of a vertical bar between columns. For example, for the letter-a data set the vertical bar indicates that the natural distribution falls between the 2% and 5% distributions (from Table 2.4 we see it is 3.9%).

Dataset	Error Rate when using Specified Training Distribution (training distribution expressed as % minority)														Relative % Improvement	
	Nat-Prune	Nat	2	5	10	20	30	40	50	60	70	80	90	95	best vs. nat	best vs. bal
letter-a	2.80 x	2.78	2.86	2.75	**2.59**	3.03	3.79	4.53	5.38	6.48	8.51	12.37	18.10	26.14	6.8	**51.9**
pendigits	3.65 +	3.74	5.77	3.95	3.63	**3.45**	3.70	3.64	4.02	4.48	4.98	5.73	8.83	13.36	7.8	**14.2**
abalone	10.68 x	10.46	**9.04**	9.61	10.64	13.19	15.33	20.76	22.97	24.09	26.44	27.70	27.73	33.91	13.6	**60.6**
sick-euthyroid	4.46 x	**4.10**	5.78	4.82	4.69	4.25	5.79	6.54	6.85	9.73	12.89	17.28	28.84	40.34	0.0	**40.1**
connect-4	10.68 x	10.56	**7.65**	8.66	10.80	15.09	19.31	23.18	27.57	33.09	39.45	47.24	59.73	72.08	**27.6**	**72.3**
optdigits	4.94 x	4.68	8.91	7.01	4.05	3.05	2.83	**2.79**	3.41	3.87	5.15	5.75	9.72	12.87	**40.4**	**18.2**
covertype	5.12 x	5.03	5.54	5.04	**5.00**	5.26	5.64	5.95	6.46	7.23	8.50	10.18	13.03	16.27	0.6	**22.6**
solar-flare	19.16 +	19.98	**16.54**	17.52	18.96	21.45	23.03	25.49	29.12	30.73	33.74	38.31	44.72	52.22	**17.2**	**43.2**
phone	12.63 x	12.62	13.45	12.87	**12.32**	12.68	13.25	13.94	14.81	15.97	17.32	18.73	20.24	21.07	**2.4**	**16.8**
letter-vowel	11.76 x	**11.63**	15.87	14.24	12.53	11.67	12.00	12.69	14.16	16.00	18.68	23.47	32.20	41.81	0.0	**17.9**
contraceptive	31.71 x	30.47	**24.09**	24.57	25.94	30.03	32.43	35.45	39.65	43.20	47.57	54.44	62.31	67.07	**20.9**	**39.2**
adult	17.42 x	17.25	18.47	17.26	**16.85**	17.09	17.78	18.85	20.05	21.79	24.08	27.11	33.00	39.75	**2.3**	**16.0**
splice-junction	8.30 +	8.37	20.00	13.95	10.72	8.68	8.50	**8.15**	8.74	9.86	9.85	12.08	16.25	21.18	2.6	6.8
network2	27.13 x	26.67	27.37	25.91	25.71	**25.66**	26.94	28.65	29.96	32.27	34.25	37.73	40.76	37.72	3.8	**14.4**
yeast	26.98 x	26.59	29.08	28.61	27.51	**26.35**	26.93	27.10	28.80	29.82	30.91	35.42	35.79	36.33	0.9	**8.5**
network1	27.57 +	27.59	27.90	27.43	26.78	**26.58**	27.45	28.61	30.99	32.65	34.26	37.30	39.39	41.09	3.7	**14.2**
car	9.51 x	8.85	23.22	18.58	14.90	10.94	8.63	8.31	7.92	**7.35**	7.79	8.78	10.18	12.86	**16.9**	7.2
german	33.76 x	33.41	**30.17**	30.39	31.01	32.59	33.08	34.15	37.09	40.55	44.04	48.36	55.07	60.99	**9.7**	**18.7**
breast-wisc	7.41 x	6.82	20.65	14.04	11.00	8.12	7.49	6.82	**6.74**	7.30	6.94	7.53	10.02	10.56	1.2	0.0
blackjack	28.14 +	**28.40**	30.74	30.66	29.81	28.67	28.56	28.45	28.71	28.91	29.78	31.02	32.67	33.87	0.0	1.1
weather	33.68 +	33.69	38.41	36.89	35.25	33.68	**33.11**	33.43	34.61	36.69	38.36	41.68	47.23	51.69	**1.7**	**4.3**
bands	32.26 +	32.53	38.72	35.87	35.71	34.76	33.33	**32.16**	32.68	33.91	34.64	39.88	40.98	40.80	1.1	1.6
market1	26.71 x	26.16	34.26	32.50	29.54	26.95	26.13	26.05	**25.77**	26.86	29.53	31.69	36.72	39.90	1.5	0.0
crx	20.99 x	**20.39**	35.99	30.86	27.68	23.61	20.84	20.82	21.48	21.64	22.20	23.98	28.09	32.85	0.0	**5.1**
kr-vs-kp	1.25 +	1.39	12.18	6.50	3.20	2.33	1.73	**1.16**	1.22	1.34	1.53	2.55	3.66	6.04	16.5	4.9
move	27.54 +	28.57	46.13	42.10	38.34	33.48	30.80	28.36	**28.24**	29.33	30.21	31.80	36.08	40.95	1.2	0.0

Table 7.1: The Effect of Training-Set Class Distribution on Error Rate

The error rate values that are not significantly different, statistically, from the lowest error rate (i.e., the comparison yields a t-test value > .10) are shaded. Thus, for the letter-a data set, the optimum range includes those class distributions that include between 2% and 10% minority-class examples—which includes the natural distribution. The last two columns in Table 7.1 show the relative improvement in error rate achieved by using the best distribution instead of the natural and balanced distributions. When this improvement is statistically significant (i.e., is associated with a t-test value ≤ .10) then the value is displayed in bold.

The results in Table 7.1 show that for 9 of the 26 data sets we are confident that the natural distribution is not within the range of optimal class distributions. For most of

these 9 data sets, using the best distribution rather than the natural distribution yields a remarkably large decrease in error rate. This provides sufficient evidence to conclude that for accuracy, when the training-set size must be limited, it is not appropriate to assume that the natural distribution is best for learning. Inspection of the error-rate results in Table 7.1 also shows that the best distribution does not differ from the natural distribution in any consistent manner—sometimes it includes more minority-class examples (e.g., *optdigits*, *car*) and sometimes fewer (e.g., *connect-4*, *solar-flare*). However, it is clear that for data sets with a substantial amount of class imbalance (the ones in the top half of the table), a balanced class distribution also is not the best class distribution for training, to minimize undifferentiated error rate. More specifically, none of the top 12 most skewed data sets have the balanced class distribution within their respective optimal ranges, and for these data sets the relative improvements over the balanced distributions are striking.

Let us now consider the error-rate values for the remaining 17 data sets for which the t-test results do not permit us to conclude that the best observed distribution truly outperforms the natural distribution. In these cases the error rate values for the 12 training-set class distributions usually form a unimodal, or nearly unimodal, distribution. This is the distribution one would expect if the accuracy of a classifier progressively degrades the further it deviates from the best distribution. This suggests that "adjacent" class distributions may indeed produce classifiers that perform differently, but that our statistical testing is not sufficiently sensitive to identify these differences. Based on this, we suspect that many of the observed improvements shown in the last column of Table 7.1 that are not deemed to be significant statistically are nonetheless meaningful.

Figure 7.1 shows the behavior of the learned classifiers for the *adult*, *phone*, *covertype*, and *letter-a* data sets in a graphical form. In this figure the natural distribution is denoted by the "X" tick mark and the associated error rate is noted above the marker. The error rate for the best distribution is underlined and displayed below the corresponding data point (for these four data sets the best distribution happens to include 10% minority-class examples).

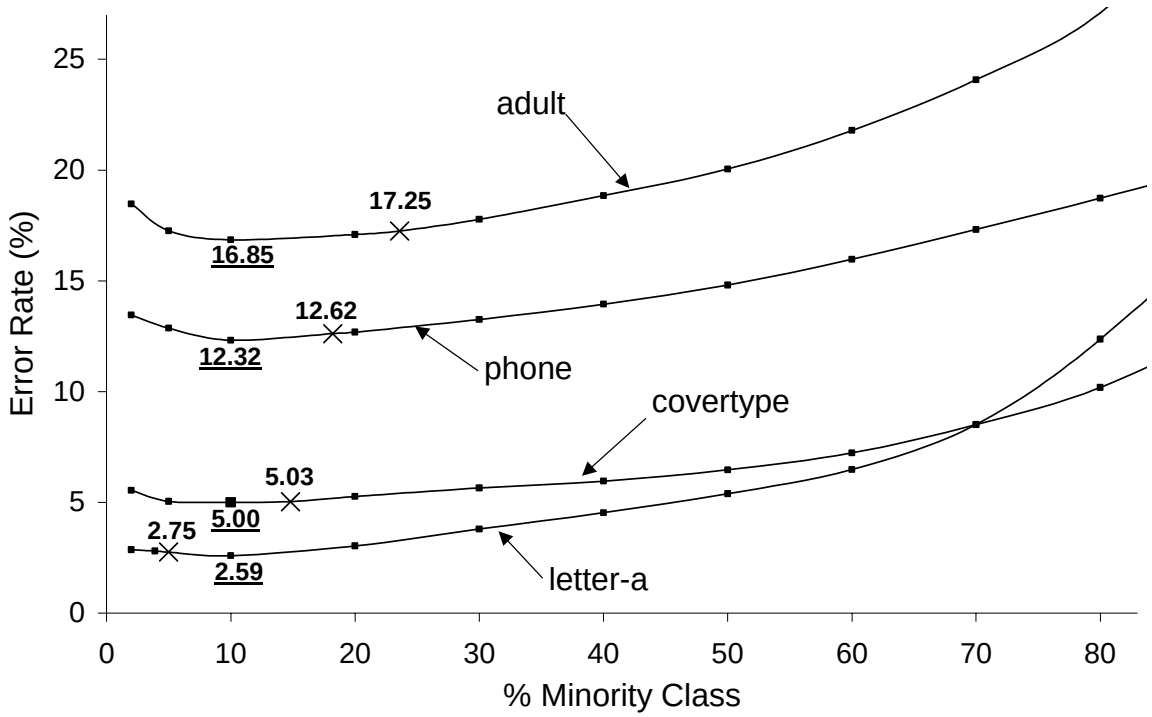

Figure 7.1: The Effect of Class Distribution on Error Rate

Note that two of the curves are associated with data sets (*adult*, *phone*) for which we are >90% confident that the best distribution performs better than the natural distribution, while for the other two curves (*covertype*, *letter-a*) we are not. Note that all four curves are perfectly unimodal. It is also clear that near the distribution that minimizes error rate, changes to the class distribution yield only modest changes in the error rate—far more

dramatic changes occur elsewhere. This is also evident for most data sets in Table 7.1. This is a convenient property given the common goal of minimizing error rate. This property would be far less evident if the corrected probability estimates, described in Chapter 6, were not used to label the classifier, since then classifiers induced from class distributions deviating from the naturally occurring distribution would be improperly biased.

Finally, to assess whether pruning would yield improved performance, consider the second column in Table 7.1, which displays the error rates that result from using C4.5 with pruning on the natural distribution. Recall that this is the only situation for which C4.5's pruning strategy will give unbiased results, since in this case the training and test sets both have the same class distribution. A "+"/"x" in the second column indicates that C4.5 with pruning outperforms/underperforms C4.5 without pruning, when learning from the natural distribution. Note that C4.5 with pruning *underperforms* C4.5 without pruning for 17 of the 26 data sets, which leads us to conclude that C4.5 without pruning is a reasonable learner. Furthermore, in no case does C4.5 with pruning generate a classifier within the optimal range when C4.5 without pruning does not also generate a classifier within this range.

7.3 The Effect of Class Distribution on AUC

The performance of the induced classifiers, using AUC as the performance measure, is displayed in Table 7.2. When viewing these results, recall that for AUC *larger* values indicate improved performance. The relative improvement in classifier performance is again specified in the last two columns, but now the relative improvement in performance is calculated in terms of the area above the ROC curve (i.e., 1 – AUC). The area above

the ROC curve is used because it better reflects the relative improvement—just as in Table 7.1 relative improvement is specified in terms of error rate rather than accuracy. As before, the relative improvements are shown in bold only if we are more than 90% confident that they reflect a true improvement in performance (i.e., t-test value $\leq .10$).

Dataset	AUC when using Specified Training Distribution (training distribution expressed as % minority)														Relative % Improv. (1-AUC)	
	Nat-prune	Nat	2	5	10	20	30	40	50	60	70	80	90	95	best vs. nat	best vs. bal
letter-a	.500 x	.772	.711	.799	.865	.891	.911	.938	.937	.944	.951	**.954**	.952	.940	**79.8**	**27.0**
pendigits	.962 x	.967	.892	.958	.971	.976	.978	**.979**	**.979**	.978	.977	.976	.966	.957	36.4	0.0
abalone	.590 x	.711	.572	.667	.710	.751	.771	.775	.776	**.778**	.768	.733	.694	.687	**25.8**	0.9
sick-euthyroid	.937 x	.940	.892	.908	.933	.943	.944	.949	.952	.951	**.955**	.945	.942	.921	**25.0**	6.3
connect-4	.658 x	.731	.664	.702	.724	.759	.763	.777	.783	**.793**	**.793**	.789	.772	.730	**23.1**	4.6
optdigits	.659 x	.803	.599	.653	.833	.900	.924	.943	.948	.959	.967	.965	**.970**	.965	**84.8**	**42.3**
covertype	.982 x	**.984**	.970	.980	**.984**	**.984**	.983	.982	.980	.978	.976	.973	.968	.960	0.0	**20.0**
solar-flare	.515 x	.627	.614	.611	.646	.627	.635	.636	.632	.650	**.662**	.652	.653	.623	**9.4**	**8.2**
phone	.850 x	.851	.843	.850	.852	.851	.850	.850	.849	.848	.848	.850	**.853**	.850	1.3	2.6
letter-vowel	.806 +	.793	.635	.673	.744	.799	.819	.842	.849	.861	**.868**	**.868**	.858	.833	**36.2**	**12.6**
contraceptive	.539 x	.611	.567	.613	.617	.616	.622	.640	.635	.635	.640	**.641**	.627	.613	**7.7**	1.6
adult	.853 +	.839	.816	.821	.829	.836	.842	.846	.851	.854	.858	**.861**	**.861**	.855	**13.7**	**6.7**
splice-junction	.932 +	.905	.814	.820	.852	.908	.915	.925	.936	.938	.944	**.950**	.944	.944	**47.4**	21.9
network2	.712 +	.708	.634	.696	.703	.708	.705	.704	.705	.702	.706	.710	**.719**	.683	3.8	4.7
yeast	.702 x	.705	.547	.588	.650	.696	**.727**	.714	.720	.723	.715	.699	.659	.621	10.9	2.5
network1	.707 +	.705	.626	.676	.697	.709	.709	.706	.702	.704	.708	**.713**	.709	.696	2.7	3.7
car	.931 +	.879	.754	.757	.787	.851	.884	.892	.916	.932	.931	**.936**	.930	.915	**47.1**	**23.8**
german	.660 +	.646	.573	.600	.632	.615	.635	**.654**	.645	.640	.650	.645	.643	.613	2.3	2.5
breast-wisc	.951 x	.958	.876	.916	.940	.958	.963	**.968**	.966	.963	.963	.964	.949	.948	23.8	5.9
blackjack	.682 x	.700	.593	.596	.628	.678	.688	.712	.713	**.715**	.700	.678	.604	.558	**5.0**	0.7
weather	.748 +	.736	.694	.715	.728	.737	.738	**.740**	.736	.730	.736	.722	.718	.702	1.5	1.5
bands	.604 x	**.623**	.522	.559	.564	.575	.599	.620	.618	.604	.601	.530	.526	.536	0.0	1.3
market1	.815 +	.811	.724	.767	.785	.801	.810	.808	.816	**.817**	.812	.805	.795	.781	3.2	0.5
crx	.889 +	.852	.804	.799	.805	.817	.834	.843	.853	.845	.857	.848	.853	**.866**	9.5	8.8
kr-vs-kp	.996 x	.997	.937	.970	.991	.994	.997	**.998**	**.998**	**.998**	.997	.994	.988	.982	33.3	0.0
move	.762 +	.734	.574	.606	.632	.671	.698	.726	.735	.738	**.742**	.736	.711	.672	3.0	2.6

Table 7.2: The Effect of Training-Set Class Distribution on AUC

In general, the optimum ranges appear to be centered near, but slightly to the right, of the balanced class distribution. For 12 of the 26 data sets the optimum range does not include the natural distribution (i.e., the third column is not shaded). Note that for these

data sets, with the exception of the *solar-flare* data set, the class distributions within the optimal range contain more minority-class examples than the natural class distribution. Based on these results we conclude even more strongly for AUC (i.e., for cost-sensitive classification and for ranking) than for accuracy that it is not appropriate simply to choose the natural class distribution for training. Table 7.2 also shows that, unlike for accuracy, a balanced class distribution generally performs very well, although it does not always perform optimally. In particular, we see that for 19 of the 26 data sets the balanced distribution is within the optimal range. This result is not too surprising since AUC, unlike error rate, is unaffected by the class distribution of the test set, and effectively factors in classifier performance over *all* class distributions.

If we look at the results with pruning, we see that for 15 of the 26 data sets C4.5 with pruning underperforms C4.5 without pruning. Thus, with respect to AUC, C4.5 without pruning is a reasonable learner. However, note that for the car data set the natural distribution with pruning falls into the optimum range, whereas without pruning it does not.

Figure 7.2 shows how class distribution affects AUC for the *adult*, *covertype*, and *letter-a* data sets (the *phone* data set is not displayed as it was in Figure 7.1 because it would obscure the *adult* data set). Again, the natural distribution is denoted by the "X" tick mark. The AUC for the best distribution is underlined and displayed below the corresponding data point. In this case we also see that near the optimal class distribution the AUC curves tend to be flatter, and hence less sensitive to changes in class distribution.

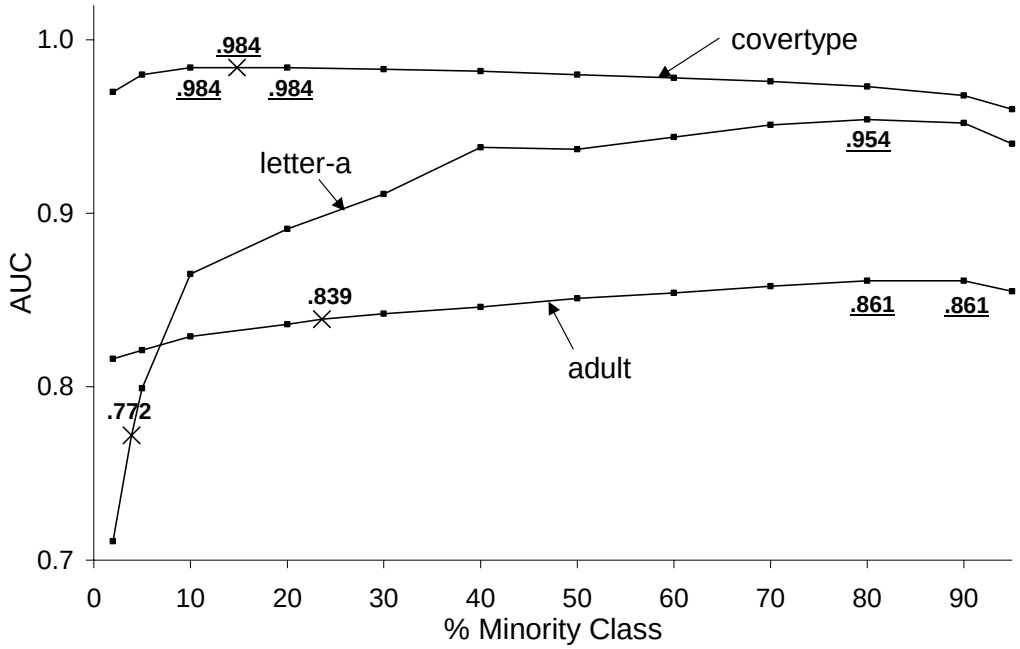

Figure 7.2: The Effect of Class Distribution on AUC

Figure 7.3 shows several ROC curves associated with the letter-vowel data set. These curves each were generated from a single run of C4.5 (this is why the AUC values do not exactly match the values in Table 7.2).

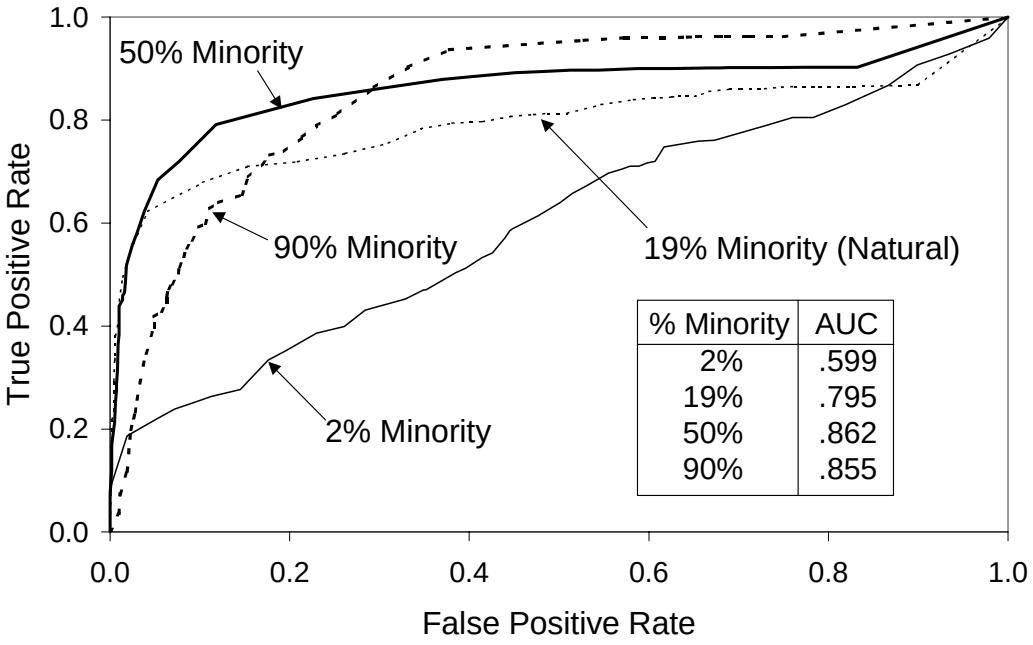

Figure 7.3: ROC Curves for the Letter-Vowel Data Set

Observe that different training distributions perform better in different areas of ROC space. Specifically note that the classifier trained with 90% minority-class examples performs substantially better than the classifier trained with the natural distribution for high true-positive rates and that the classifier training with 2% minority-class examples performs fairly well for low true-positive rates. These results are easily explained. With only a small sample of minority-class examples (2%) a classifier can identify only a few minority-labeled "rules" with high confidence. However, with a much larger sample of minority-class examples (90%) it can identify many more such minority-labeled rules. However, for this data set a balanced distribution has the largest AUC and performs best overall. One interesting thing to note is that the curve generated using the balanced class distribution almost always outperforms the curve associated with the natural distribution (for low false-positive rates the natural distribution performs slightly better).

7.4 The Interaction between Class Distribution and Training-Set Size

Experiments were run to establish the relationship between training-set size, class distribution and classifier performance for the *phone*, *covertype*, and *adult* data sets. These three data sets were selected because they are all quite large, and hence the amount of training data can be dramatically reduced while still yielding meaningful results.

The results are summarized in Table 7.3. Classifier performance is again reported for thirteen class distributions, but this time also using the following nine training-set sizes, expressed in terms of the fraction of available training data: 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2, 3/4, and 1. The natural class distribution is denoted by an asterisk and, for each training-set size, the class distribution that yields the best performance is displayed in bold and is underlined.

Data Set	Size	Metric	2	5	10	18.2*	20	30	40	50	60	70	80	90	95
PHONE	1/128		.641	.737	.784	**.793**	.792	.791	.791	.789	.788	.786	.785	.774	.731
|	1/64		.707	.777	.784	**.803**	.803	**.803**	.801	.802	.801	.798	.799	.788	.744
|	1/32		.762	.794	.809	**.812**	**.812**	.811	.811	.811	.810	**.812**	.811	.805	.778
|	1/16		.784	.813	.816	.823	.823	**.824**	.818	.821	.822	.821	.822	.817	.805
|	1/8	AUC	.801	.823	.828	.830	.830	.830	.830	.829	.830	.829	**.831**	.828	.818
|	1/4		.819	.835	.837	.839	.839	.837	.837	.836	.836	.836	.838	.836	.832
|	1/2		.832	.843	**.846**	**.846**	.845	.845	.843	.843	.843	.843	.844	**.846**	.844
|	3/4		.838	.847	.849	.849	.849	.848	.846	.847	.846	.847	.848	**.851**	.848
|	1		.843	.850	.852	.851	.851	.850	.850	.849	.848	.848	.850	**.853**	.850
|	1/128		17.47	16.42	**15.71**	16.10	16.25	17.52	18.81	21.21	22.87	26.40	30.43	33.26	37.27
|	1/64		17.01	15.75	15.21	**15.12**	15.20	16.39	17.59	19.60	22.11	24.80	27.34	30.21	26.86
|	1/32		16.22	15.02	14.52	**14.50**	14.75	15.41	16.81	18.12	20.02	21.77	24.86	25.31	28.74
|	1/16	Error	15.78	14.59	**14.01**	14.02	14.18	14.70	16.09	17.50	18.68	20.70	22.46	24.15	24.52
|	1/8	Rate	15.17	14.08	**13.46**	13.61	13.71	14.27	15.30	16.51	17.66	19.66	21.26	23.23	23.33
|	1/4		14.44	13.55	**13.12**	13.23	13.27	13.85	14.78	15.85	17.09	18.94	20.43	22.28	22.90
|	1/2		13.84	13.18	**12.81**	12.83	12.95	13.47	14.38	15.30	16.43	17.88	19.57	21.68	21.68
|	3/4		13.75	13.03	**12.60**	12.70	12.74	13.35	14.12	15.01	16.17	17.33	18.82	20.43	21.24
|	1		13.45	12.87	**12.32**	12.62	12.68	13.25	13.94	14.81	15.97	17.32	18.73	20.24	21.07
			2	**5**	**10**	**20**	**23.9***	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**95**
ADULT	1/128		.571	.586	.633	.674	.680	.694	.701	.704	.723	.727	**.728**	.722	.708
|	1/64		.621	.630	.657	.702	.714	.711	.722	.732	.739	.746	**.755**	.752	.732
|	1/32		.638	.674	.711	.735	.742	.751	.755	.766	.762	.765	**.772**	.766	.759
|	1/16		.690	.721	.733	.760	.762	.778	.787	.791	**.794**	.787	.785	.780	.771
|	1/8	AUC	.735	.753	.768	.785	.787	.793	.799	.809	.812	**.816**	.813	.803	.797
|	1/4		.774	.779	.793	.804	.809	.813	.820	.827	.831	.832	**.834**	.824	.811
|	1/2		.795	.803	.812	.822	.825	.829	.834	.838	.841	.847	**.849**	.847	.834
|	3/4		.811	.814	.823	.830	.833	.837	.843	.845	.849	.853	**.856**	.855	.848
|	1		.816	.821	.829	.836	.839	.842	.846	.851	.854	.858	**.861**	**.861**	.855
|	1/128		23.80	23.64	**23.10**	23.44	23.68	23.90	25.22	26.94	29.50	33.08	37.85	46.13	48.34
|	1/64		23.32	22.68	22.21	**21.77**	21.80	23.08	24.38	26.29	28.07	31.45	36.41	43.64	47.52
|	1/32		22.95	22.09	21.12	**20.77**	20.97	21.11	22.37	24.41	27.08	30.27	34.04	42.40	47.20
|	1/16	Error	22.66	21.34	20.29	**19.90**	20.07	20.37	21.43	23.18	25.27	28.67	33.41	40.65	46.68
|	1/8	Rate	21.65	20.15	19.13	**18.87**	19.30	19.67	20.86	22.33	24.56	27.14	31.06	38.35	45.83
|	1/4		20.56	19.08	**18.20**	18.42	18.70	19.12	20.10	21.39	23.48	25.78	29.54	36.17	43.93
|	1/2		19.51	18.10	**17.54**	**17.54**	17.85	18.39	19.38	20.83	22.81	24.88	28.15	34.71	41.24
|	3/4		18.82	17.70	**17.17**	17.32	17.46	18.07	18.96	20.40	22.13	24.32	27.59	33.92	40.47
|	1		18.47	17.26	**16.85**	17.09	17.25	17.78	18.85	20.05	21.79	24.08	27.11	33.00	39.75
			2	**5**	**10**	**14.8***	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**95**
COVERTYPE	1/128		.767	.852	.898	.909	**.916**	.913	**.916**	**.916**	.909	.901	.882	.854	.817
|	1/64		.836	.900	.924	.932	**.937**	.935	.936	.932	.928	.922	.913	.885	.851
|	1/32		.886	.925	.942	.947	**.950**	.947	.948	.948	.944	.939	.930	.908	.876
|	1/16		.920	.944	.953	.957	**.959**	**.959**	**.959**	.957	.955	.951	.945	.929	.906
|	1/8	AUC	.941	.955	.963	.965	.967	.968	**.969**	.968	.967	.963	.957	.948	.929
|	1/4		.953	.965	.970	.973	.975	.976	.975	.973	.972	.970	.965	.956	.943
|	1/2		.963	.972	.979	**.981**	**.981**	.980	.978	.977	.975	.972	.970	.961	.953
|	3/4		.968	.976	.982	.982	**.983**	.982	.980	.979	.976	.975	.971	.966	.958
|	1		.970	.980	**.984**	**.984**	**.984**	.983	.982	.980	.978	.976	.973	.968	.960
|	1/128		**10.44**	10.56	10.96	11.86	13.50	16.16	18.26	20.50	23.44	26.95	31.39	37.92	44.54
|	1/64		9.67	**9.29**	10.23	11.04	12.29	14.55	16.52	18.58	21.40	24.78	27.65	34.12	41.67
|	1/32		8.87	**8.66**	9.44	10.35	11.29	13.59	15.34	17.30	19.31	21.82	24.86	28.37	33.91
|	1/16	Error	8.19	**7.92**	8.93	9.67	10.37	11.93	13.51	15.35	17.42	19.40	22.30	25.74	28.36
|	1/8	Rate	7.59	**7.32**	7.87	8.65	9.26	10.31	11.63	13.06	14.68	16.39	18.28	22.50	26.87
|	1/4		6.87	**6.44**	7.04	7.49	8.01	9.05	9.86	10.56	11.45	12.28	14.36	18.05	22.59
|	1/2		6.04	**5.71**	5.97	6.45	6.66	7.14	7.53	8.03	8.80	9.94	11.44	14.85	18.37
|	3/4		5.81	**5.31**	5.48	5.75	5.87	6.25	6.57	6.89	7.58	8.72	10.69	13.92	16.29
|	1		5.54	5.04	**5.00**	5.03	5.26	5.64	5.95	6.46	7.23	8.50	10.18	13.03	16.27

Table 7.3: The Effect of Training-Set Size and Class Distribution on Learning

The results in Table 7.3 show that while the position of the best class distribution varies somewhat with training-set size, in many cases, especially with error rate, the variation is small. This gives support to the notion that there is a "best" marginal class distribution for a learning task. The results also indicate that, for any fixed class distribution, increasing the size of the training set *always* leads to improved classifier performance. Note that classifier performance has not reached a plateau for any of the three data sets, for either error rate or AUC. This is important because if a plateau had been reached (i.e., learning had stopped), then it would be possible to reduce the size of the training set without degrading classifier performance. Because this is not the case, the results in Table 7.3 indicate that, for these three data sets (and C4.5), it may be profitable to select the training examples carefully when forming the training set. This provides one practical motivation for the research on class distribution described in this thesis.

Six comparisons are highlighted in Table 7.3, by using a line to connect pairs of data points. For each of these six cases, competitive or improved performance is achieved from fewer training examples. That is, in each of these six cases, the data point corresponding to the smaller data-set size performs as well or better than the data point that corresponds to the larger data-set size (the latter being either the natural distribution or a balanced one). As an example, consider the adult data set and the AUC performance metric. Table 7.3 shows that one can achieve a better AUC value (.849 vs. .839) by using, instead of all available training data and the natural class distribution, one-half of the available training data with a class distribution that includes 80% minority-class examples. As a second example, one can achieve a competitive error rate on the phone data set (12.60% vs. 12.62%) by using ¾ of the available data rather than the full data set, if

one uses a class distribution that includes 10% minority-class examples rather than the natural class distribution.

Figures 7.4 and 7.5 provide a graphical representation of the data, to show how class distribution and training-set size interact to affect classifier performance for the *adult* data set. In these figures each performance curve is associated with a single training-set size. Because the performance curves always improve with increasing data set size, only the curves corresponding to the smallest and largest training-set sizes are explicitly labeled.

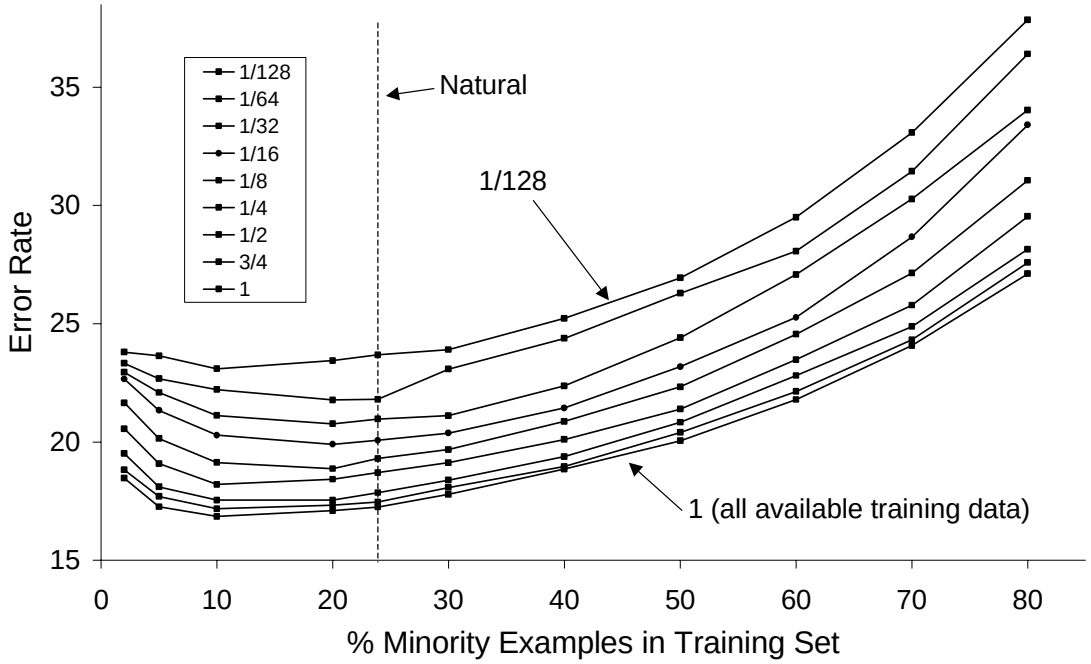

Figure 7.4: Effect of Class Distribution and Training-Set Size on Error Rate

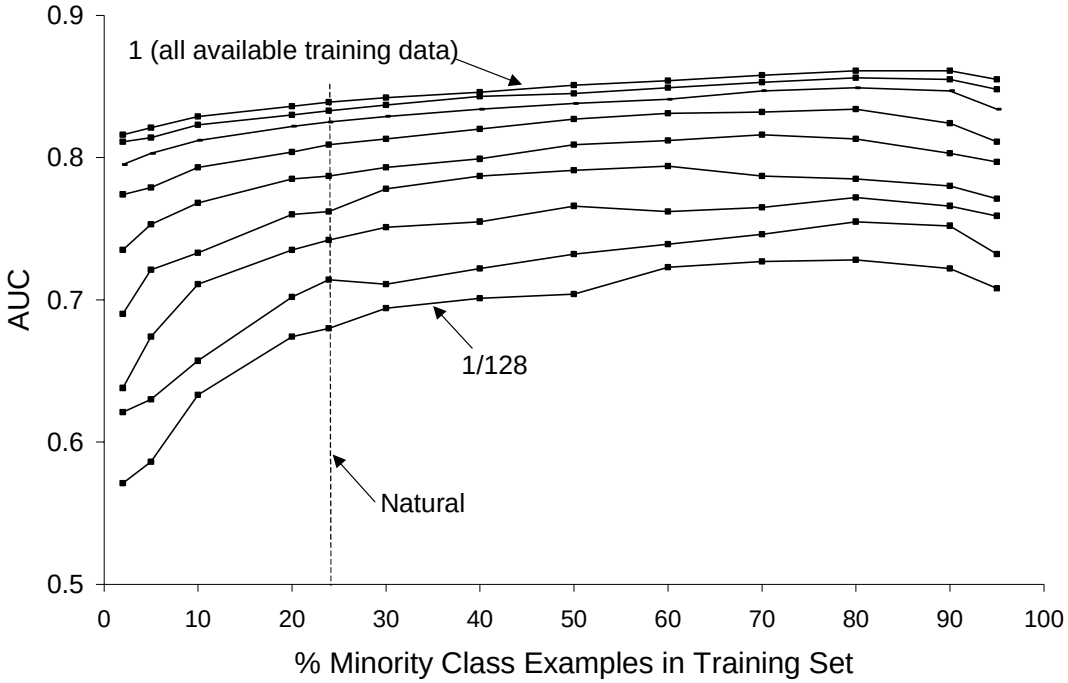

Figure 7.5: Effect of Class Distribution and Training-Set Size on AUC

In these figures, for a given x-value, the space between successive y-values indicates the change in performance due to changes in training-set size. Based on this it is clear that classifier performance does begin to flatten out as more training data become available, but that it does not flatten out completely. Also note that as the training-set size increases the choice of class distribution becomes somewhat less important, since the range of error-rate values for each curve diminishes.

7.5 Summary

This chapter demonstrated how the choice of class distribution affects classifier performance. The experimental results show that when accuracy is used to measure classifier performance, the class distributions that yield the best classifiers tend to occur near the naturally occurring class distribution, but that significant improvements in classifier performance may still occur by using a different class distribution. When AUC is used to

measure classifier performance, a balanced class distribution generally performs quite well—and almost always performs better than the naturally occurring class distribution. The results from experiments that vary the amount of training data show that, as hoped, one can achieve competitive classifier performance using fewer training examples if the class distribution is chosen carefully.

Chapter 8

A Budget-Sensitive Algorithm for Selecting Training Data

"You must choose, but choose wisely"

- Indiana Jones and the Last Crusade

The results from the previous chapter demonstrate that some marginal class distributions yield classifiers that perform substantially better than others. Unfortunately, forming the thirteen training sets of size *n*, each with a different class distribution, requires nearly 2*n* examples (2*n* examples would be needed if the class distribution ranged from 100% minority-class examples to 100% majority-class examples). When it is costly to obtain training examples in a form suitable for learning, then this approach is not feasible—and ultimately is self-defeating. Ideally, one would like to select exactly *n* training examples, have them all used in the training set, and have the resulting class distribution yield a classifier that performs better than one generated from any other class distribution. In this chapter we describe and evaluate a heuristic, budget-sensitive, progressive-sampling algorithm for selecting training data that approximates this ideal.

8.1 Budget-Sensitive Sampling

Usable training examples, as described earlier, are sometimes costly to obtain. In these cases, the cost associated with forming the training set may need to be limited. Let us assume that this total cost must not exceed some budgeted value, B. In this chapter we

assume that the cost of procuring usable training examples is uniform and does not differ based on the class of the example. Thus the budget B will make it possible to procure some number of examples, *n*. In the remainder of this chapter the budget is expressed with respect to the number of examples that may be procured, rather than directly in terms of cost (e.g., dollars).

The budget-sensitive sampling strategy makes two additional assumptions. First, it assumes that the number of potentially available training examples from each class is sufficiently large so that a training set with *n* examples can be formed with any desired marginal class distribution. The second assumption is that the cost of executing the learning algorithm is negligible compared to the cost of procuring examples. This assumption permits the learning algorithm to be run multiple times, in order to provide guidance about which examples to select.

Formally, given a budget B that permits *n* examples to be procured, a budget-sensitive sampling strategy will select no more than *n* examples. The goal of the sampling strategy is to select *x* minority-class examples and *y* majority-class examples, where $x + y = n$, such that the resulting distribution yields a classifier that performs well (i.e., near optimally). We say such a strategy is *budget-efficient* if all examples that are selected are used in the final training set, when forming the "desired" (in this case heuristically determined) class distribution.

8.2 The Budget-Sensitive Progressive Sampling Algorithm

This section describes the budget-sensitive progressive sampling algorithm. The algorithm begins with a small amount of training data and progressively adds training examples using a geometric sampling schedule (Provost, Jensen & Oates, 1999). The propor-

tion of minority-class examples and majority-class examples added in each iteration of the algorithm is determined empirically by forming several class distributions from the currently available training data, evaluating the classification performance of the resulting classifiers, and then determining the class distribution that performs best. The algorithm implements a beam-search through the space of possible class distributions.

The sampling algorithm will be guaranteed to be budget-efficient. The key to ensuring this is to constrain the search through the space of class distributions, so that all examples obtained during one iteration of the algorithm are guaranteed to be used in subsequent iterations. Note, however, that the heuristically determined class distribution associated with the final training set is not guaranteed to yield the best-performing classifier; however, as we will show, the classifier induced using this class distribution performs well in practice.

8.2.1 Description of the Algorithm

The algorithm is outlined in Table 8.1, using pseudo-code. This is followed by a line-by-line explanation of the algorithm. The algorithm takes three user-specified input parameters: μ, the geometric factor used to determine the rate at which the training-set size grows, *n*, the budget, and *cmin*, the minimum fraction of minority-class examples and majority-class examples that are assumed to appear in the final training set in order for the budget-efficiency guarantee to hold.[5] For the results presented in this section, μ is set to 2, so that the training-set size doubles every iteration of the algorithm, and *cmin* is set to 1/32.

[5] Consider the degenerate case where the algorithm determines that the best class distribution contains no minority-class examples or no majority-class examples. If the algorithm begins with even a single example of this class, then it will not be efficient.

1. minority = majority = 0; # current number minority/majority examples
2. $K = \left\lceil \log_{\mu}\left(\frac{1}{c \min}\right)\right\rceil$; # number of iterations is K+1
3. for (j = 0; j ≤ K; j = j+1) # for each iteration (e.g., j = 0, 1,2,3,4)
4. {
5. size = $n(1/\mu)^{K-j}$ # set training-set size for iteration j
6. if (j = 0)
7. beam_bottom = 0; beam_top = 1;
8. else
9. $\text{beam\_radius} = \frac{\min(best, 1-best)}{\mu - 1/\mu + 1}$
10. beam_bottom = best – beam_radius; beam_top = best + beam_radius;
11. min_needed = size • beam_top; # number minority examples needed
12. maj_needed = size • (1.0 – beam_bottom); # number majority examples needed
13. if (min_needed > minority)
14. request (min_needed - minority) additional minority-class examples;
15. if (maj_needed > majority)
16. request (maj_needed - majority) additional majority-class examples;
17. if (j ≠ K)
18. eval(beam_bottom, beam_top, size); # evaluate distributions in the beam
19. else
20. eval(best, best, size); # last iteration just evaluate previous best
21. }

Table 8.1: A Budget-Sensitive Progressive Sampling Algorithm

The algorithm begins by initializing the values for the *minority* and *majority* variables, which represent the total number of minority-class examples and majority-class examples requested by the algorithm. Then, in line 2, the number of iterations of the algorithm is determined, such that the initial training-set size, which is subsequently set in line 5, will be at most $cmin \bullet n$. This will allow all possible class distributions to be formed using at most *cmin* minority-class examples and *cmin* majority-class examples. For example, given that μ is 2 and *cmin* is 1/32, in line 2 variable *K* will be set to 5 and in line 5 the initial training-set size will be set to 1/32 *n*.

Next, in lines 6-10, the algorithm determines the class distributions to be considered in the current iteration by setting the boundaries of the beam. For the first iteration, all class distributions are considered (i.e., the fraction of minority-class examples in the training set may vary between 0 and 1). In subsequent iterations, the beam will be centered on the class distribution that performed best in the previous iteration. In line 9 the radius of the beam is set such that the ratio beam_top/beam_bottom will equal μ. For example, if μ is 2 and *best* is .15, then *beam_radius* is .05 and the beam will span from .10 to .20—which differ by a factor of 2 (i.e., μ).

In lines 11 and 12 the algorithm computes the number of minority-class examples and majority-class examples needed to form the class distributions that fall within the beam. These values are determined from the class distributions at the boundaries of the beam. In lines 13-16 additional examples are requested, if required. In lines 17-20 an evaluation procedure is called to form the class distributions within the beam and then to induce and to evaluate the classifiers. At a minimum this procedure will evaluate the class distributions at the endpoints and at the midpoint of the beam; however, this procedure may be

implemented to evaluate additional class distributions within the beam. The procedure will set the variable *best* to the class distribution that performs best. If the best performance is achieved by several class distributions, then a resolution procedure is needed. The resolution procedure used in this thesis is to select the class distribution for which the surrounding class distributions perform best; if this still does not yield a unique value, then the class distribution closest to the center of the beam is chosen. In any event, for the last iteration, only one class distribution is evaluated—the class distribution from the previous iteration that yields the best performance. This is necessary to ensure budget-efficiency, since if more than one class distribution were evaluated in this case, then some examples would need to be discarded to form the best-performing class distribution.

8.2.2 Proof of Budget-Efficiency

This algorithm is *guaranteed* to request only examples that are subsequently used in the final training set, which will have the heuristically determined class distribution. This guarantee can be verified inductively. First, the base case. The calculation for K in line 2 ensures that the initial training set will contain $cmin \bullet n$ training examples. Since we assume that the final training set will have at least *cmin* minority-class examples and *cmin* majority-class examples, all examples used to form the initial training set are guaranteed to be included in the final training set. Note that *cmin* may be set arbitrarily small—the smaller *cmin* the larger K and the smaller the size of the initial training set.

The inductive step is based on the observation that because the radius of the beam in line 9 is set so that the beam spans at most a factor of μ, all examples requested in each iteration are guaranteed to be used in the final training set. To see this, we will work backward from the final iteration (but the reasoning works both ways). Assume that the

result of the algorithm is that the fraction of minority-class examples in the final training set is p, so that there are $p \bullet n$ minority-class examples in the final training set. This means that p was the best distribution from the previous iteration. Since p must fall somewhere within the beam for the previous iteration and the beam must span a factor μ, we can say the following: the fraction of minority-class examples in the previous iteration could range from p/μ (if p was at the top of the previous beam) to $\mu \bullet p$ (if p was at the bottom of the previous beam). Since the previous iteration contains n/μ examples, due to the geometric sampling scheme, then the previous iteration has at most $(\mu \bullet p) \bullet n/\mu$, or $p \bullet n$, minority-class examples. Thus, in all possible cases all minority-class examples from the previous iteration can be used in the final interaction. This argument applies similarly to the majority-class examples and can be extended backwards to previous iterations.[6] Thus, because of the bound on the initial training-set size and the restriction on the width of the beam not to exceed the geometric factor μ, the algorithm guarantees that all examples requested during the execution of the algorithm will be used in the final training set.

8.2.3 Minor Modifications to the Algorithm to Simplify Evaluation

The experimental results in Table 7.3 from Chapter 7 show how class distribution and training-set size affect classifier performance. These results are based on thirteen class distribution values. So that we can use these results to evaluate the sampling algorithm, the sampling algorithm must be modified so that when the beam is set in lines 6-10 of Table 8.1, only these thirteen values are chosen for evaluation. This is done by setting the low end (high end) of the beam to the class distribution listed in Table 7.3 that is just

[6] The only exception is for the first iteration of the algorithm, since in this situation the beam is unconditionally set to span all class distributions. This is the reason why the *cmin* value is required to provide the efficiency guarantee.

below (above) the best performing class distribution. As an example, if the best performing class distribution contains 30% minority-class examples, then the bottom of the beam is set to include 20% minority-class examples and the top of the beam to include 40% minority-class examples. Although this will sometimes allow the beam to span a range greater than μ (2) and hence eliminates the guarantee of budget-efficiency, as we shall see, in practice this does not result in a problem (i.e., the algorithm is still budget-efficient).

In addition, one slight improvement was made to the algorithm. Specifically, for any iteration, if the number of examples already in hand (procured in previous iterations) is sufficient to evaluate additional class distributions, then the beam is widened to include these additional class distributions. This can occur because during the first iteration the beam is set very wide.

8.2.4 A Detailed Example

This section provides a detailed iteration-by-iteration example describing the sampling algorithm as it is applied to the phone data set, when error rate is used to evaluate classifier performance. However, before providing the detailed description, we preview the main results by showing the "trajectory" of the sampling algorithm as it searches for a good class distribution. The trajectory is depicted graphically in Figure 8.1 (the trajectory is based on the *class-distr* values listed shortly in Table 8.2).

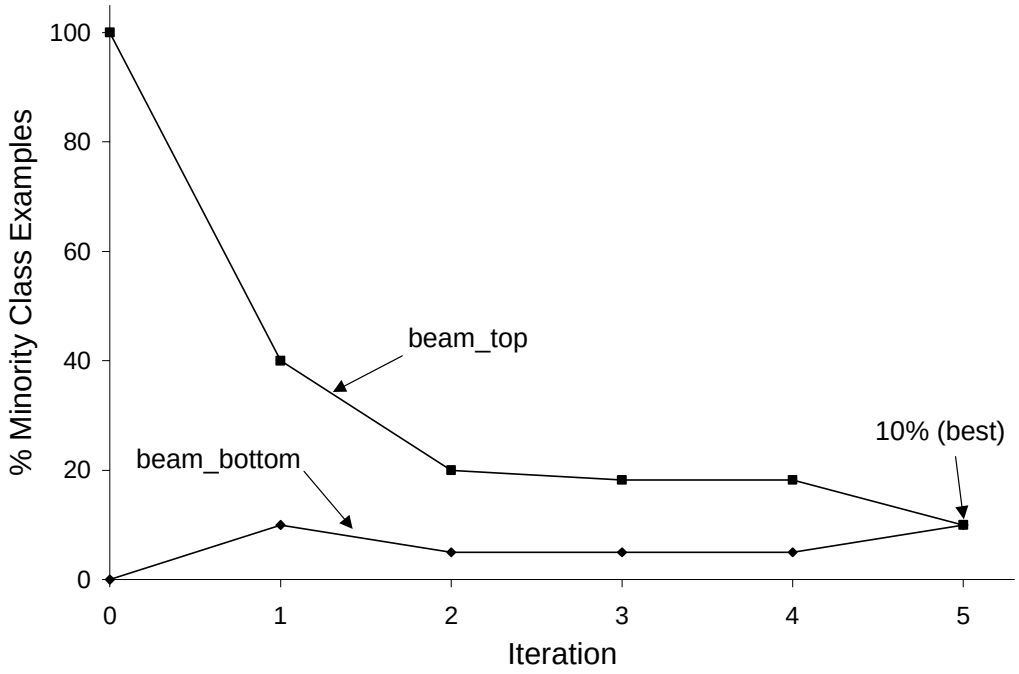

Figure 8.1: Trajectory of the Sampling Algorithm

In the following description, note that budget refers to the budget expended thus far. Hence a value of .5*n* means that half of the budgeted examples have been procured. All experimental results (including the ones presented in the next section) are based on a geometric factor, μ, of 2, and a value of *cmin* of 1/32. The total budget available for procuring training examples is *n*. Based on these values, the value of *K*, which determines the number of iterations of the algorithm and is computed on line 2 of Table 8.1, is set to 5. Finally, note that since we use the values from Table 7.3 to evaluate the sampling algorithm, the number of budgeted training examples to be procured, *n*, corresponds to the number of training examples used when generating Table 7.3.

The detailed iteration-by-iteration description of the sampling algorithm, as it is applied to the phone data set with error rate as the performance measure, is shown below.

A more concise summary is then provided in Table 8.2, which records the values of the key variables during each iteration.

---

j = 0 Training-set size = 1/32 *n*. Form 13 data sets, which will contain between 2% and 95% minority-class examples. This requires .0298*n* *(*95% of 1/32 *n)* minority-class examples and .0307*n* (100%-2% = 98% of 1/32 *n*) majority-class examples. Induce and then evaluate the resulting classifiers. Based on the results in Table 7.3, the natural distribution, which contains 18.2% minority-class examples, performs best. Total Budget: .0605*n* (.0298*n* minority, .0307*n* majority).

j = 1 Training-set size = 1/16 *n*. Form data sets corresponding to the best-performing class distribution form the previous iteration (18.2% minority) and the adjoining class distributions used in the beam search, which contain 10% and 20% minority-class examples. This requires .0250*n* (20% of 1/16 *n*) minority-class examples and .0563*n* (90% of 1/16 *n*) majority-class examples. Since .0298*n* minority-class examples were previously obtained, class distributions containing 30% and 40% minority-class examples can also be formed without requesting additional examples. This iteration requires .0256*n* additional majority-class examples. The best-performing distribution contains 10% minority-class examples. Total Budget: .0861 *n* (.0298*n* minority, .0563*n* majority).

j = 2 Training-set size = 1/8 *n*. Since the 10% distribution performed best, the beam search evaluates the 5%, 10%, and 18.2% minority-class distributions. The 20% class distribution is also evaluated since this requires only .0250*n* of the .0298*n* previously obtained minority-class examples. A total of .1188*n* (95% of 1/8 *n*) majority-class examples are required. The best performing distribution contains 10% minority-class examples. This iteration requires .0625*n* additional majority-class examples. Total Budget: .1486*n* (298*n* minority, 1188*n* majority).

j = 3 Training-set size = 1/4 *n*. The distributions to be evaluated are 5%, 10%, and 18.2%. There are no “extra” minority-class examples available to evaluate addi-

tional class distributions. This iteration requires .0455*n* (18.2% of 1/4 *n*) minority-class examples and .2375*n* (95% of 1/4 *n*) majority-class examples. The best-performing class distribution contains 10% minority-class examples. Total Budget: .2830*n* (.0455*n* minority, .2375*n* majority)

j = 4 Training-set size = 1/2 *n*. The 5%, 10%, and 18.2% class distributions are evaluated. This iteration requires .0910*n* (18.2% of 1/2 *n*) minority-class examples and .4750*n* (95% of 1/2 *n*) majority-class examples. The best-performing distribution contains 10% minority-class examples. Total Budget: .5660*n* (.0910*n* minority, .4750*n* majority).

j = 5 Training-set size = *n*. For this last iteration only the best class distribution from the previous iteration is evaluated. Thus, a data set of size *n* is formed, containing .1*n* minority-class examples and .9*n* majority-class examples. Thus .0090*n* additional minority-class examples and .4250*n* additional majority-class examples are required. Since all the previously obtained examples are used, there is no "waste" and the budget is not exceeded. Total Budget: 1.0*n* (.1000*n* minority, .9000*n* majority).

This detailed description is summarized in Table 8.2. Note that the total budget is not exceeded since the budget used in 1*n*.

				Expressed as a fraction on *n*				
j	size	class-distr	best	min-need	maj-need	minority	majority	budget
0	1/32 *n*	all	18.2%	.0298	.0307	.0298	.0307	.0605
1	1/16 *n*	10, *18.2*, 20, 30, 40	10%	.0250	.0563	.0298	.0563	.0861
2	1/8 *n*	5, *10*, 18.2, 20	10%	.0250	.1188	.0298	.1188	.1486
3	1/4 *n*	5, *10*, 18.2	10%	.0455	.2375	.0455	.2375	.2830
4	1/2 *n*	5, *10*, 18.2	**10%**	.0910	.4750	.0910	.4750	.5660
5	1 *n*	10		.1000	.9000	.1000	.9000	1.0000

Table 8.2: Compact Description of the Execution of the Sampling Algorithm

8.3 Experimental Results

The budget-sensitive progressive-sampling algorithm, using the results from Table 7.3, was applied to the phone, adult, and covertype data sets using both error rate and AUC to measure classifier performance. The execution of the sampling algorithm on these data sets is described in Table 8.3 using the compact tabular notation introduced in the Table 8.2.

Data set	Metric	j	size	class-distr	best	min-need	maj-need	minority	majority	budget
Phone	ER	0	1/32 *n*	all	18.2	.0298	.0307	.0298	.0307	.0605
		1	1/16 *n*	10, *18.2*, 20, 30, 40	10	.0250	.0563	.0298	.0563	.0861
		2	1/8 *n*	5, *10*, 18.2, 20	10	.0250	.1188	.0298	.1188	.1486
		3	1/4 *n*	5, *10*, 18.2	10	.0455	.2375	.0455	.2375	.2830
		4	1/2 *n*	5, *10*, 18.2	**10**	.0910	.4750	.0910	.4750	.5660
		5	1 *n*	10		.1000	.9000	.1000	.9000	1.0
Phone	AUC	0	1/32 *n*	all	20	.0298	.0307	.0298	.0307	.0605
		1	1/16 *n*	18.2, *20*, 30, 40	30	.0250	.0511	.0298	.0511	.0809
		2	1/8 *n*	20, *30*, 40	30	.0500	.1000	.0500	.1000	.1500
		3	1/4 *n*	20, *30*, 40	20	.1000	.2000	.1000	.2000	.3000
		4	1/2 *n*	18.2, *20*, 30	**18.2**	.1500	.4090	.1500	.4090	.5590
		5	1 *n*	18.2		.1820	.8180	.1820	.8180	1.0
Adult	ER	0	1/32 *n*	all	20	.0298	.0307	.0298	.0307	.0605
		1	1/16 *n*	10, *20*, 23.9, 30, 40	20	.0250	.0563	.0298	.0563	.0861
		2	1/8 *n*	10, *20*, 23.9	20	.0299	.1125	.0299	.1125	.1424
		3	1/4 *n*	10, *20*, 23.9	10	.0598	.2250	.0598	.2250	.2848
		4	1/2 *n*	5, *10*, 20	**20**	.1000	.4750	.1000	.4750	.5750
		5	1 *n*	20		.2000	.8000	.2000	.8000	1.0
Adult	AUC	0	1/32 *n*	all	80	.0298	.0307	.0298	.0307	.0605
		1	1/16 *n*	60, 70, *80*, 90	70	.0563	.0250	.0563	.0307	.0870
		2	1/8 *n*	60, *70*, 80	70	.1000	.0500	.1000	.0500	.1500
		3	1/4 *n*	60, *70*, 80	80	.2000	.1000	.2000	.1000	.3000
		4	1/2 *n*	70, *80*, 90	**80**	.4500	.1500	.4500	.1500	.6000
		5	1 *n*	80		.8000	.2000	.8000	.2000	1.0
Covertype	ER	0	1/32 *n*	all	5	.0298	.0307	.0298	.0307	.0605
		1	1/16 *n*	2, *5*, 10, 20, 30, 40	5	.0250	.0613	.0298	.0613	.0911
		2	1/8 *n*	2, *5*, 10, 20	5	.0250	.1225	.0298	.1225	.1523
		3	1/4 *n*	2, *5*, 10	5	.0250	.2450	.0298	.2450	.2748
		4	1/2 *n*	2, *5*, 10	**5**	.0500	.4900	.0500	.4900	.5400
		5	1 *n*	5		.0500	.9500	.0500	.9500	1.0
Covertype	AUC	0	1/32 *n*	all	20	.0298	.0307	.0298	.0307	.0605
		1	1/16 *n*	14.8, *20*, 30, 40	30	.0250	.0533	.0298	.0533	.0831
		2	1/8 *n*	20, *30*, 40	40	.0500	.1000	.0500	.1000	.1500
		3	1/4 *n*	30, *40*, 50	30	.1250	.1750	.1250	.1750	.3000
		4	1/2 *n*	20, *30*, 40	**20**	.2000	.4000	.2000	.4000	.6000
		5	1 *n*	20		.2000	.8000	.2000	.8000	1.0

Table 8.3: Complete Summary Description of the Execution of the Algorithm

The results in Table 8.3 indicate that in no case is the budget exceeded, which means that all examples requested during the execution of the algorithm are used in the final training set, with the heuristically-determined class distribution (i.e., the algorithm is budget-efficient).

The performance of the sampling algorithm is summarized in Table 8.4, along with the performance of three other strategies for selecting the class distribution of the training data. These additional strategies are: 1) always pick the natural class distribution, 2) always pick the balanced class distribution, and 3) pick the class distribution that performs best over the thirteen class distributions in Table 7.3. Table 8.4 also specifies the cost for each strategy, which is based on the number of training examples requested by the algorithm. This cost is expressed with respect to the budget *n* (each strategy yields a final training set with *n* examples).

	Sampling Algorithm			Pick Natural			Pick Balanced			Pick Best		
Data Set	**ER**	**AUC**	**Cost**	**ER**	**AUC**	**Cost**	**ER**	**AUC**	**Cost**	**ER**	**AUC**	**Cost**
phone	12.32%	0.851	*n*	12.32%	0.851	*n*	14.81%	0.849	*n*	12.32%	0.853	1.93*n*
adult	17.09%	0.861	*n*	17.25%	0.839	*n*	20.05%	0.851	*n*	16.85%	0.861	1.93*n*
covertype	5.04%	0.984	*n*	5.03%	0.984	*n*	6.46%	0.980	*n*	5.00%	0.984	1.93*n*

Table 8.4: Comparative Performance of the Sampling Algorithm

We begin by discussing the costs associated with each of the four selection strategies. The strategies that involve picking the natural or balanced class distributions require exactly *n* examples to be selected and hence are budget-efficient. The "Pick Best" strategy, which evaluates thirteen class distributions using between 2% and 95% minority-class examples, requires that .95*n* minority-class examples and .98*n* majority-class examples be chosen. This yields a total cost of 1.93*n* and hence is not budget-efficient. Unlike these three "fixed" strategies, the cost of the budget-sensitive sampling algorithm de-

pends on the performance of the induced classifiers. Even though this algorithm, with the changes described in Section 8.2.3, is not guaranteed to be budget-efficient, it nonetheless is in all cases—since for both error rate and AUC the cost incurred is always exactly *n*.

Next we compare the results for classifier performance in Table 8.4. These results show that by using the budget-sensitive progressive-sampling algorithm to choose the training data it is possible to achieve results that are no worse and sometimes better than the strategies of always using the natural or balanced class distributions, without requiring that any extra examples to be procured. Specifically, note that in no case does the strategy of always using the balanced distribution outperform the sampling algorithm and that in only one case (for covertype using error rate) does the strategy of using the natural distribution outperform the sampling algorithm (and in this case the difference is minimal). The budget-sensitive algorithm does almost as well (essentially as well in 5 of 6 cases) as the "Pick Best" strategy, which is almost twice as costly. Based on these results, the budget-sensitive progressive-sampling algorithm is attractive—it incurs the minimum possible cost in terms of procuring examples while permitting the class distribution for training to be selected using some intelligence.

8.4 Summary

This chapter introduced a budget-sensitive progressive sampling algorithm that uses some intelligence in determining the class distribution used for learning, while selecting the minimum number of examples possible (i.e., no examples are wasted). A slightly modified version of the basic algorithm was evaluated on three very large data sets and shown to perform nearly as well as the optimal strategy, which always chooses the best class distribution for learning. The sampling algorithm described in this chapter shows that the

experimental results from earlier chapters can be put to practical use—that in practice one can determine a good class distribution for learning by intelligently selecting the training examples.

Chapter 9

Research Related to Class Distribution

This second part of this thesis focused on class distribution, its effect on decision-tree learning and how choosing the class distribution carefully when training data is costly can compensate for a limited amount of training data. This chapter describes related research, which may be placed into the following categories: research concerning the relationship between class distribution and classifier performance, research on how to reduce the amount of necessary labeled training data, research on progressive sampling strategies, and research on how to learn when data sets have highly unbalanced class distributions. Research in each of these categories is described in turn.

9.1 The Relationship between Class Distribution and Classifier Performance

Several researchers have considered the question of what class distribution to use for a fixed training-set size, and, more generally, how class distribution affects classifier performance. Both Catlett (1991) and Chan and Stolfo (1998) study the relationship between (marginal) training class distribution and classifier performance when the training-set size is held fixed. However, these studies focus most of their attention on other issues and have two main weaknesses with respect to their study of class distribution. First, both of these studies analyze only a few data sets, which makes it impossible to draw general conclusions about the relationship between class distribution and classifier per-

formance. Secondly, these studies fail to adjust for changes made to the class distribution of the training set, as described in Chapter 6.

Chan & Stolfo (1998) show, based on three data sets, that when accuracy is the performance metric, a training set that uses the natural class distribution yields the best results. These results agree partially with our results—although we show that the natural distribution does not always maximize accuracy, we do show that the optimal distribution generally is close to the natural distribution. The different conclusions are likely due to the fact the Chan & Stolfo do not adjust for the changes made to the class distribution of the training data. Not adjusting for these changes, as shown in this thesis, will negatively affect classifier performance when the class distribution is modified, and hence will bias the results to favor the naturally occurring (unmodified) class distribution.

Chan & Stolfo also show that when actual costs are factored in (i.e., the cost of a false positive is not the same as a false negative), the natural distribution does not perform best; rather a training distribution closer to a balanced distribution performs best. They also observe, as we did, that by increasing the percentage of minority-class examples in the training set, the induced classifier performs better at classifying minority examples.

9.2 Reducing the Need for Labeled Training Data

Several researchers have looked at the general question of how to reduce the need for labeled training data by selecting the data intelligently, but without explicitly considering the class distribution. For example, Cohn et al. (1994) and Lewis and Catlett (1994) use "active learning" to add examples to the training set for which the classifier is least certain about the classification. Saar-Tsechansky and Provost (2001, 2003) provide an overview of such methods and also extend them to cover AUC and other non-accuracy based

performance metrics. However, the setting where these methods are applicable is different from the setting we consider. In particular, these methods assume either that arbitrary examples can be labeled or that the descriptions of a pool of unlabeled examples are available and the critical cost is associated with labeling them (so the algorithms select the examples intelligently rather than randomly). In our typical setting, the cost is in procuring the descriptions of the examples—the labels are known beforehand.

9.3 Progressive Sampling Strategies

The budget-sensitive sampling strategy described in Chapter 8 employs a type of progressive sampling. There has been some prior research on progressive sampling strategies. John and Langley (1996) show how one can use the extrapolation of learning curves to determine when classifier performance using a subset of available training data comes close to the performance that would be achieved by using the full data set. Provost et al. (1999) suggest using a geometric sampling schedule and show that it is often more efficient than using all of the available training data. The techniques described by John and Langley (1996) and Provost et al. (1999) do not change the distribution of examples in the training set, but rather rely on taking random samples from the available training data. Our progressive sampling routine extends these methods by stratifying the sampling by class and using the information acquired during the process to select a good final class distribution.

9.4 Handling Highly Unbalanced Class Distributions

There has been a considerable amount of research on how to build “good” classifiers when the class distribution of the data is highly unbalanced and it is costly to misclassify

minority-class examples (Japkowicz et al., 2000). This research is related to the research in this thesis because a frequent approach for learning from highly skewed data sets is to modify the class distribution of the training set. Under these conditions, classifiers that optimize for accuracy are especially inappropriate because they tend to generate trivial models that almost always predict the majority class. A common approach for dealing with highly unbalanced data sets is to reduce the amount of class imbalance in the training set. This tends to produce classifiers that perform better on the minority class than if the original distribution were used. Note that in this situation the training-set size is not fixed and the motivation for changing the distribution is usually to produce a "better" classifier—not to reduce, or minimize, the training-set size.

The two basic methods for reducing class imbalance in training data are under-sampling and over-sampling. In this context, under-sampling discards examples in the majority class while over-sampling replicates examples in the minority class (Breiman, et al. 1984; Kubat & Matwin, 1997; Japkowicz & Stephen, 2001). Neither approach consistently outperforms the other nor does any specific under-sampling or over-sampling rate consistently yield the best results. Estabrooks and Japkowicz (2001) address this issue by showing that a mixture-of-experts approach, which combines classifiers built using under-sampling and over-sampling methods with various sampling rates, can produce consistently good results.

Both under-sampling and over-sampling have known drawbacks. Under-sampling throws out potentially useful data while over-sampling *increases the size of the training set* and hence the time to build a classifier. Furthermore, since most over-sampling methods make exact copies of minority class examples, overfitting is likely to occur—

classification rules may be induced to cover a single replicated example. Recent research has focused on improving these basic methods. Kubat and Matwin (1997) employ an under-sampling strategy that intelligently removes majority examples by removing only those majority examples that are "redundant" or that "border" the minority examples—figuring they may be the result of noise. Chawla et al. (2000) combine under-sampling and over-sampling methods, and, to avoid the overfitting problem, form new minority class examples by interpolating between minority-class examples that lie close together. Chan and Stolfo (1998) take a somewhat different, and innovative, approach. They first run preliminary experiments to determine the best class distribution for learning and then generate multiple training sets with this class distribution. This is typically accomplished by including all minority-class examples and some of the majority-class examples in each training set. They then apply a learning algorithm to each training set and then combine the generated classifiers to form a composite learner. This method ensures that all available training data are used, since each majority-class example will be found in at least one of the training sets.

The research in this thesis can properly be viewed as research into under-sampling and its effect on classifier performance. However, given this perspective, our research performs under-sampling in order to reduce the training-set size, whereas in the research relating to skewed data sets the primary motivation is to improve classifier performance. For example, Kubat and Matwin (1997) motivate the use of under-sampling because "… adding examples of the majority class to the training set can have a detrimental effect on the learner's behavior." A consequence of these different motivations is that in our experiments we discard examples belonging to the minority and majority classes (although

in different proportions), while in the research concerned with learning from skewed distributions it is only majority-class examples that are discarded.

The use of under-sampling for reducing the training-set size (and thereby reducing cost) may be the more practically useful perspective. Reducing the class imbalance in the training set effectively causes the learner to impose a greater cost for misclassifying minority-class examples (Breiman et al., 1984). Thus, when the cost of acquiring and learning from the data is not an issue, cost-sensitive or probabilistic learning methods are a more direct and arguably more appropriate way of dealing with class imbalance, because they do not have the problems, noted earlier, that are associated with under-sampling and over-sampling. Such approaches have been shown to outperform under-sampling and over-sampling (Japkowicz & Stephen, 2002). To quote one of the papers that considers this issue, "All of the data available can be used to produce the tree, thus throwing away no information, and learning speed is not degraded due to duplicate instances" (Drummond & Holte 2000, page 239).

9.5 Summary

This chapter reviewed research related to the class distribution work described in this thesis. The research most directly related to our research, by Chan and Stolfo (1998), produced similar results regarding the relationship between class distribution and classifier performance. The observed differences are likely due to the fact that Chan and Stolfo did not correct for changes made to the class distribution of the training data. This chapter also reviewed research concerned with intelligently selecting data when data is costly—research that use criteria other than the class label to select the data. Progressive sampling strategies were also reviewed. Much of this chapter focused on how the class dis-

tribution of highly skewed data sets may be modified by under-sampling or over-sampling the data. The relationship between these techniques, and the research in this thesis, was discussed.

Chapter 10

Conclusions and Future Work

"What we call the beginning is often the end.
And to make an end is to make a beginning.
The end is where we start from."

- T. S. Eliot, "Four Quartets"

"A conclusion is the place where you got tired of thinking."

- Martin H. Fischer

This thesis provides an empirical study of small disjuncts and class distribution and their impact on decision-tree learning. This chapter summarizes the main contributions and lessons learned from these two studies. Limitations with these studies are then discussed, including the reliance on decision tree learning throughout this thesis. In this discussion we do provide some reasons to believe that our results may generalize beyond this one important class of learners. Areas of possible future research are then discussed, followed by some final remarks.

10.1 Summary of Contributions

Although the study of small disjuncts and the study of class distribution are separate, many of the main contributions for each are related, since the studies share the same general framework. We begin by describing these common contributions.

This thesis provides the most comprehensive empirical studies to date of the role that small disjuncts and class distribution play in learning. Whereas previous research studies, for both topics, have only analyzed a handful of data sets, in this thesis we analyze a large set of benchmark data sets (thirty data sets for small disjuncts and twenty-six for class distribution). Because of this, we are able to quantify the impact that small disjuncts and class distribution have on decision tree learning, note patterns of behavior and draw conclusions from the results.

Each study began by measuring the impact and role of the relevant phenomena (small disjuncts, class distribution) on learning under normal circumstances (e.g., without adding noise or altering the class distribution). For the study of small disjuncts, a new metric, error concentration, was used to summarize the distribution of errors with respect to disjunct size. This new metric made it possible, for the first time, to compare these distributions across classifiers. The experimental results demonstrated that the errors are highly concentrated in the small disjuncts for most classifiers, but that there are a substantial number of classifiers for which this is not the case.

Analysis of these small disjunct results indicates that classifiers with relatively low error rates almost always have high error concentrations while this is not true of classifiers with high error rates. This analysis further indicates that this pattern is due to the fact that classifiers with low error rates generally contain some very accurate large disjuncts. We conclude from this that concepts that can be learned well tend to contain very large subconcepts and that C4.5 and Ripper generate classifiers with similar error concentrations because both are able to form accurate large disjuncts to cover these large subconcepts.

Similarly, the effect that class distribution has on decision tree learning under normal circumstances, when the class distribution of the training data is not altered, was also measured. These results were analyzed to look for any differences between the minority and majority classes. The results show that the minority-class predictions consistently have a much higher error rate than majority-class predictions and that minority-class test examples are misclassified much more often than majority-class test examples. The difference in behavior for the predictions was attributed to the fact that there are more majority-class examples in the test data (the "test distribution effect") and that minority class examples tend to be covered by smaller, more error prone, disjuncts. The difference in behavior for the test examples was attributed to the fact that accuracy favors the majority class—false negative errors are preferred over false positive errors.

Each of the studies next made changes to the experimental setup, to gain additional insight into the learning process. For the small disjunct study, factors such as pruning, training-set size, noise, and class imbalance were varied. The goal of these experiments was twofold: to provide a better understanding of the role of small disjuncts in decision tree learning and to provide a better understanding of decision tree learning in general. The experiments and subsequent analysis yielded many interesting observations, only some of which are highlighted below.

Pruning was shown to eliminate many small disjuncts, thereby reducing the error concentration of the induced classifiers. However, even with pruning most classifiers had error concentrations that were decidedly positive, indicating that pruning does not totally eliminate, or, more properly, *mask*, the problem with small disjuncts. Pruning was shown to be most effective when the unpruned classifier has a high error concentration, indicat-

ing that most errors are concentrated in the smaller disjuncts. Pruning was also shown to be only moderately effective as a strategy for addressing the problem with small disjuncts. Finally, this thesis pointed out a weakness of pruning—it distributes the errors more uniformly throughout the disjuncts. We show that because of this pruning hurts the accuracy of large disjuncts and will generally degrade classification performance when only a subset of the available examples need to be classified.

The analysis of training-set size shows that an increase in the amount of training data almost always leads to an increase in error concentration. This change occurs because as more training data is made available, the large disjuncts, which may cover the areas in the target concept that are expressible using axis-parallel cuts in the instance space, can be learned more accurately. In contrast, small disjuncts, which may be used to approximate the portions of the target concept that cannot be expressed using axis-parallel cuts (and generally lie close to the decision boundary), will still contain some errors. Furthermore, the additional training data may cause some small subconcepts to be sampled for the first time, resulting in small disjuncts. These disjuncts will tend to be error prone because with few training examples, it will be difficult to accurately determine the correct boundaries of the subconcept.

Noise was investigated to determine its effect on small disjuncts and error concentration. Noisy data was shown to cause many erroneous small disjuncts to be formed and to break down highly accurate large disjuncts. Pruning was shown to combat the effects of noise by dramatically reducing the number of small disjuncts formed due to the noisy data. This was shown to help preserve the accuracy of the induced classifier. The increase in error rate that does occur with pruning is due to the inability of the pruning

strategy to distinguish between small subconcepts of the target concept and noise. Supporting this is the fact that noise appears to have less of a deleterious effect on learning when the classifier induced without noise has a low error concentration—indicating that the classifier may be made up primarily of large disjuncts.

Finally, class imbalance was analyzed to see how it affects the distribution of errors by disjunct size. Changing the class distribution of the data set to remove class imbalance was shown to consistently reduce the error concentration of the induced classifier, indicating that class imbalance is partly responsible for the problem with small disjuncts. The reason that class imbalance leads to an increase in error concentration is as follows. Minority-class examples are more difficult to classify than majority-class examples due to the test distribution effect (i.e., because there are more majority-class test examples). Because our results show that small disjuncts are more likely to be labeled with the minority class than large disjuncts, small disjuncts therefore should have a higher error rate than large disjuncts. This says that part of the problem with small disjuncts is related to class imbalance.

Next, we return to the contributions from the study of class distribution. The class distribution of the training data was altered to study the effect of class distribution on classification performance. Because this alteration would unduly bias the classifier to favor the preferentially sampled class, the class probability estimates produced by the classifier were adjusted and the class labels re-assigned based on these new estimates. The method for correcting these estimates was described in detail in this thesis. Experimental results showed for the first time that adjusting the probability estimates yields a substantial improvement in classifier accuracy. This indicates that the results from past research

on class distribution, which did not adjust the classifier to account for changes in class distribution, are suspect. In particular, these results will tend to be biased toward the natural class distribution.

The experimental results using C4.5 indicate that when accuracy is the performance measure, the best class distribution for learning tends to be near the natural distribution—although the use of a class distribution other than the naturally occurring distribution often leads to substantial improvements in classifier performance. The results also indicate that when AUC is the performance measure, then a balanced class distribution will generally perform quite well, although not always optimally. Therefore, we recommend, as a general guideline, that if the relative costs of misclassifying examples are not known, or the true, underlying, class distribution is not known, that one consider using a balanced class distribution for learning when the amount of training data must be limited.

However, one can improve upon this general guideline by interleaving data procurement and learning, using feedback from learning to guide the search for a good class distribution. This method is feasible when data can be incrementally procured by class or when the amount of data is limited due to costs associated with learning from the data (e.g., computational limitations). In this latter case, labeled data are already available, so procurement of examples by class is trivial. Based on the empirical evidence presented in this thesis, our budget-sensitive progressive-sampling algorithm almost always performs as well or better than the strategies of always choosing the natural distribution or a balanced distribution—and never does considerably worse. Furthermore, this sampling algorithm yields classifiers that nearly perform as well as those generated from the optimal class distribution. Based on these results, the budget-sensitive progressive-sampling

algorithm is attractive—it incurs the minimum possible cost in terms of procuring examples while permitting the class distribution for training to be selected using some intelligence.

Machine learning and data mining practitioners often need to make changes to the class distribution of training data, but have had little guidance in the past on how to do this. Changes to the class distribution are seldom done in a principled manner and the reasons for changing the distribution—and the consequences—are often not understood. The research presented in this thesis should provide these practitioners with a better understanding of the relationship between class distribution and classifier performance, the issues involved, and permits them to learn more effectively when there is a need to limit the amount of training data.

10.2 Limitations and Future Work

This section describes limitations with the research described in this thesis and also suggests possible avenues for future research. The first limitation we discuss concerns the reliance on decision tree learning (and rule learning for the small disjuncts study). This class of learners was selected because of its popularity and importance. We believe that the reliance on decision tree learning is not as large a concern for the class distribution study as for the study of small disjuncts. This is because there are some reasons to believe that the class distribution results aren't unduly tied to the learning algorithm. Namely, since the role that class distribution plays in learning—and the reasons listed in Section 6.6.2 for why a classifier will perform worse on the minority class—are not specific to decision-tree learners, one would expect other learners to behave similarly.

Nonetheless, it would certainly be worthwhile in future work to repeat the experiments for the class distribution study using additional learners.

The study of small disjuncts, however, may be much more closely tied to the type of learning algorithm. This is because small disjuncts may be formed, in part, due to the inability of a learner to express the target concept (recall the example in Figure 3.1 in which a decision tree learner attempts to learn a non-axis parallel boundary). For this reason, it would be particularly useful to extend the study of small disjuncts to include other learners—including learners such as neural networks that have greater expressive power.

One specific problem with using C4.5 to study the effect of class distribution on learning is that C4.5 does not account for changes made to the class distribution of the training data. Differences between the class distribution of the training and test data are accounted for in a post-processing step, by re-computing the probability estimates at the leaves and using these estimates to re-label the tree. There are two drawbacks with this approach. The first is that if C4.5 had knowledge of the target (i.e., test) distribution *during* the tree-building process, then a different decision tree might be constructed. The second drawback is that we could not use C4.5's pruning strategy when alterations were made to the training data, since pruning is sensitive to changes in class distribution.

It would be worthwhile in future research to analyze the effect of class distribution on learning using a learner that is sensitive to changes in distribution or cost-sensitive (changes in class distribution are effectively equivalent to changes in misclassification cost). Such a learner would generate a classifier based on full knowledge of the changes made to the class distribution and would also use this information when pruning.

We should point out, however, that even with the two drawbacks just noted, there are reasons to believe that the existing limitation with C4.5 is not that serious—and that our results would not be that different had C4.5 been able to factor in changes to the class distribution. First, Drummond and Holte (2000) showed that there are splitting criteria that are completely insensitive to differences in class distribution and that these splitting criteria perform as well or better than methods that explicitly factor in changes to class distribution. They further showed that C4.5's splitting criterion is relatively insensitive to the class distribution—and therefore to changes in class distribution. Thus, the tree produced by C4.5 might not change much even if changes in distribution were factored in. Secondly, there are reasons to believe that not using pruning did not unduly affect our results. We can say this because we showed that the observed differences in behavior between the minority and majority classes still exist with pruning and that C4.5 without pruning performs competitively with C4.5 with pruning. Moreover, other research (Bradford et al., 1998) indicates that classifier performance does not generally improve when pruning takes class distribution and costs into account.

Another limitation with our study of class distribution is that only two-class learning problems were analyzed. Although this simplified the analysis, it limited the scope of the study. It would be quite interesting to see if the observed patterns of behavior change for data sets with many classes.

The budget-sensitive progressive-sampling algorithm described in this thesis assumes that the cost of obtaining examples is the same independent of the class of the example. In practice, however, this is often not the case—examples belonging to the rarer class are often more expensive (e.g., in the telecommunication domain fraudulent charges must be

verified). It would be interesting to extend the sampling algorithm to take non-uniform procurement costs into account and see how such differences in cost affects the optimal class distribution.

This thesis showed that it is possible to improve classifier performance by changing the class distribution of the training data, so that it no longer adheres to the naturally occurring class distribution. Perhaps a similar approach can be employed to better learn small subconcepts, thereby reducing the problem with small disjuncts. The basic idea is to select additional training examples that would fall within a small disjunct, in order to "enlarge" the disjunct. Similar approaches have been developed previously, to intelligently choose examples based on feedback from the learning process, but not solely based on disjunct size. For example, as described in Chapter 9, over-sampling methods sometimes introduce additional minority-class examples, by either duplicating existing examples, or by intelligently combining existing minority-class examples. The approach of selecting additional examples is actually more feasible in the context defined in this thesis, when the amount of training data needs to be limited, because examples that fall into these small disjuncts may already exist. It is worthwhile to note the research in this thesis already incidentally addresses the idea of selecting examples based on disjunct size. This is because the recommended strategy of choosing a balanced class distribution for learning to maximize AUC eliminates many small disjuncts (i.e., those associated with the minority class). Part of the success of this strategy is therefore due to the fact that it preferentially samples examples that would otherwise be classified by small disjuncts.

The pruning results associated with the study of small disjuncts showed that pruning degrades the accuracy of the highly accurate, large disjuncts. Given that large disjuncts are often amongst the best classification rules (i.e., will have the best predictive accuracy), these intriguing results indicate that pruning may generally be harmful when one need only classify some of the available examples. Thus, these results warrant further attention. First, additional learners and pruning strategies should be analyzed to see if they exhibit the same behavior. Next, the classification rules should be ordered using criteria other than disjunct size (e.g., training accuracy) to judge the quality of the rule, to see if pruning still has a deleterious effect on the best classification rules.

Error concentration was used in this thesis to describe classifier performance and to assess how various factors (pruning, noise, etc.) affect classifier performance. A key question is, can error concentration be used as a parameter to improve learning? This is similar in spirit to the use of disjunct size, by other researchers, to determine the learner, or bias, to be used to classify an example. For example, can error concentration, measured using one learner (e.g., C4.5) be used to select the most appropriate learner? More fundamentally, which learning methods perform best for data sets associated with high error concentrations? With low error concentrations? Based on the results for pruning in this thesis, perhaps the more intriguing question is can error concentration be used to help determine how much pruning should be done?

10.3 Final Remarks

We began this thesis by discussing the value of studying those situations and portions of a classifier that are responsible for contributing most of the errors. In this thesis we saw that small disjuncts contribute many, if not most, classification errors. We also saw that

rare classes are a problem for learning. That is, we saw that the inability of a classifier to effectively classify minority-class examples leads to many, if not most, classification errors. While we proposed no specific solution for dealing with small disjuncts, we did show that one can improve learning in the presence of rare classes by carefully choosing the class distribution of the training data. For example, we suggested the use of a balanced class distribution for learning when the amount of training data must be limited and one wants to generate a classifier robust to changes in misclassification cost and class distribution (i.e., wants to maximize AUC). This thesis shows that by studying small disjuncts and class distribution—by focusing on where the errors are—one can gain useful insights into the learning process.

References

Ali, K. M. & Pazzani, M. J. (1992). Reducing the Small Disjuncts Problem by Learning Probabilistic Concept Descriptions, in T. Petsche editor, *Computational Learning Theory and Natural Learning Systems*, Volume 3, Cambridge Massachusetts: MIT Press.

Barron, A., Rissanen, J., & Yu, B. (1998). The minimum description length principle in coding and modeling. IEEE Transactions on Information Theory, 44(6), 2743-2760.

Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: bagging, boosting, and variants. *Machine Learning*, 36, 105-139.

Blake, C. L. & Merz, C. J. (1998). UCI Repository of ML Databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Dept. of Computer Science.

Bradley, A. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7), 1145-1159.

Bradford, J.P., Kunz, C., Kohavi, R., Brunk, C., & Brodley, C. E. (1998). Pruning decision trees with misclassification costs. In *Proceedings of the European Conference on Machine Learning*, pp. 131-136.

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

Carvalho, D. R. & Freitas, A. A. (2000). A hybrid decision tree/genetic algorithm for coping with the problem of small disjuncts in data mining. In *Proceedings of the 2000 Genetic and Evolutionary Computation Conference*, 1061-1068. Las Vegas, NV.

Cohen, W. (1995). Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, 115-123.

Cohen, W. & Singer, Y. (1999). A simple, fast, and effective rule learner. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 335-342. Menlo Park, Calif.: AAAI Press.

Cohn, D., Atlas, L. and Ladner, R. (1994). Improved generalization with active learning. *Machine Learning*, 15:201-221.

Catlett, J. (1991). *Megainduction: machine learning on very large databases*. Ph.D. thesis, Department of Computer Science, University of Sydney.

Chan, P., & Stolfo, S. (1998). Toward scalable learning with non-uniform class and cost distributions: a case study in credit card fraud detection. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pp. 164-168, Menlo Park, CA: AAAI Press.

Chawla, N., Bowyer, K., Hall, L., & Kegelmeyer, W. P. (2000). SMOTE: synthetic minority over-sampling technique. In *International Conference on Knowledge Based Computer Systems*.

Danyluk, A. P. & Provost, F. J. (1993). Small disjuncts in action: learning to diagnose errors in the local loop of the telephone network. In *Proceedings of the Tenth International Conference on Machine Learning*, 81-88.

Drummond, C., & Holte, R.C. (2000). Exploiting the cost (in)sensitivity of decision tree splitting criteria. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 239-246.

Elkan, C. (2001). The foundations of cost-sensitive learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence.*

Estabrooks, A., & Japkowicz, N. (2001). A Mixture-of-Experts Framework for Concept-Learning from Imbalanced Data Sets. *In Proceedings of the 2001 Intelligent Data Analysis Conference*.

Fawcett, T. & Provost, F. (1997). Adaptive Fraud Detection. *Data Mining and Knowledge Discovery* **1**(3): 291-316.

Glover, E. J., Tsioutsiouliklis, K., Lawrence, S., Pennock, D. M. & Flake, G. W. (2002). Using web structure for classifying and describing web pages. In *Proceedings of the Eleventh International World Wide Web Conference*, 562-569.

Good, I. J. (1965). *The Estimation of Probabilities*. Cambridge, MA: M.I.T. Press.

Hand, D. J. (1997). *Construction and Assessment of Classification Rules*. Chichester, UK: John Wiley and Sons.

Holte, R., C., Acker, L. E., & Porter, B. W. (1989). Concept learning and the problem of small disjuncts. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 813-818. San Mateo, CA: Morgan Kaufmann.

Japkowicz, N. & Stephen, S. (2002). The Class Imbalance Problem: A Systematic Study. *Intelligent Data Analysis Journal, 6(5).*

Japkowicz, N., Holte, R. C., Ling, C. X., & Matwin S. (Eds.) (2000). In *Papers from the AAAI Workshop on Learning from Imbalanced Data Sets*. Tech, rep. WS-00-05, Menlo Park, CA: AAAI Press.

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

John, G. H., & Langley, P. (1996). Static versus dynamic sampling for data mining. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 367-370. Menlo Park, CA. AAAI Press.

Kendall, M. & Gibbons, K. D. (1990). *Rank Correlation Methods. Oxford University Press, fifth edition.*

Kubat, M., & Matwin, S. (1997). Addressing the curse of imbalanced training sets: one-sided selection. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 179-186.

Lewis, D. D., & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pp.148-156.

Perlich, C., Provost, F., & Siminoff, J. (2003). Tree induction vs. logistic regression: a learning-curve analysis. *Journal of Machine Learning Research*. To appear.

Provost, F., Fawcett, T., & Kohavi, R. (1998). The case against accuracy estimation for comparing classifiers. In *Proceedings of the Fifteenth International Conference on Machine Learning*. San Francisco, CA: Morgan Kaufmann.

Provost, F., Jensen, D., & Oates, T. (1999). Efficient progressive sampling. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*. ACM Press.

Provost, F., & Fawcett, T (2001). Robust classification for imprecise environments. *Machine Learning*, 42, 203-231.

Provost, F., & Domingos, P. (2001). Well-trained PETs: improving probability estimation trees. CeDER Working Paper #IS-00-04, Stern School of Business, New York University, New York, NY.

Quinlan, J. R. (1986). The effect of noise on concept learning. In R. S. Michalski, J. G. Carbonell & T. M. Mitchell (eds.), *Machine Learning, an Artificial Intelligence Approach, Volume II*, 149-166, Morgan Kaufmann.

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

Quinlan, J. R. (1991). Technical note: improved estimates for the accuracy of small disjuncts. *Machine Learning*, 6(1).

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

Saar-Tsechansky, M., & Provost, F. (2001). Active learning for class probability estimation and ranking. *In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, Seattle, WA.

Saar-Tsechansky, M. and F. Provost (2003). Active Sampling for Class Probability Estimation and Ranking. To appear in *Machine Learning*.

SAS Institute (2001). Getting Started With SAS Enterprise Miner. Cary, NC: SAS Institute Inc.

Saerens, M., Latinne, P., & Decaestecker, C. (2002). Adjusting the outputs of a classifier to new *a priori* probabilities: a simple procedure. *Neural Computation*, 14:21-41.

Swets, J., Dawes, R., & Monahan, J. (2000). Better decisions through science. *Scientific American*, October 2000: 82-87.

Ting, K. M. (1994). The problem of small disjuncts: its remedy in decision trees. In *Proceedings of the Tenth Canadian Conference on Artificial Intelligence*, 91-97.

Turney P. (2000). Types of cost in inductive learning. In *Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning*, 15-21, Stanford, CA.

Van den Bosch, A., Weijters, A., Van den Herik, H. J., & Daelemans, W. (1997). When small disjuncts abound, try lazy learning: A Case Study. In *Proceedings of the Seventh Belgian-Dutch Conference on Machine Learning*, 109-118.

Weiss, G. M. (1995). Learning with rare cases and small disjuncts. In *Proceedings of the Twelfth International Conference on Machine Learning*, 558-565.

Weiss, G. M. & Hirsh, H. (1998). The problem with noise and small disjuncts. In *Proceedings of the Fifteenth International Conference on Machine Learning*, 574-578.

Weiss, G. M. & Hirsh, H. (1998). Learning to predict rare events in event sequences. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, 359-363.

Weiss, G. M. & Hirsh, H. (2000). A quantitative study of small disjuncts. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, 665-670, Austin, Texas.

Weiss, G. M. & Provost, F. (2001). The effect of class distribution on classifier learning: an empirical study. Technical Report ML-TR-44, Department of Computer Science, Rutgers University, August 2001. An updated version has been submitted to the Journal of Artificial Intelligence Research (JAIR).

Zadrozny, B., & Elkan, C. (2001). Learning and making decisions when costs and probabilities are both unknown. Tech. rep. CS2001-0664, Department of Computer Science and Engineering, University of California, San Diego.

Vita

Gary M. Weiss

Degrees

1981 Graduated from Mamaroneck High School, Mamaroneck, New York.

1985 B.S. in Computer Science from Cornell University, Ithaca, New York.

1986 M.S in Computer Science from Stanford University, Stanford, California.

2003 Ph.D. in Computer Science from Rutgers University, New Brunswick, New Jersey.

Work History

1985-present AT&T Laboratories (formerly Bell Laboratories). 1985-1991: System Software Development Group; 1991-1994: Software Development Tools Group; 1994-1995: New Product Serviceability Group; 1995-1996 Switch Maintenance Development and Integration Group; 1996-1998 Technologies Development & Deployment Group; 1998-present Market Analysis Group.

Selected Publications

1995 G. M. Weiss. Learning with Rare Cases and Small Disjuncts. In *Proceedings of the Twelfth International Conference on Machine Learning*, Morgan Kaufmann, San Francisco, CA, 558-565.

1996 D. Dvorak, A. Mishra, J. Ros, G. M. Weiss and D. Litman. R++: Using Rules in Object-Oriented Designs. In Addendum *Object-Oriented Programming Systems, Languages and Applications,* San Jose, CA.

1998 G. M. Weiss and H. Hirsh. The Problem with Noise and Small Disjuncts. *In Proceedings of the Fifteenth International Conference on Machine Learning*, Morgan Kaufmann, San Francisco, 574-578.

1998 G. M. Weiss. ANSWER: Network Monitoring Using Object-Oriented Rules. In *Proceedings of the Tenth Conference on Innovative Applications of Artificial Intelligence*, AAAI Press, 1087-1093.

1998 G. M. Weiss, J. Eddy, and S. Weiss. Intelligent Telecommunication Technologies. In *Knowledge-Based Intelligent Techniques in Industry (chapter 8)*, L. C. Jain, editor, CRC Press.

1998 G. M. Weiss and J. Ros. Implementing Design Patterns with Objected-Oriented Rules. *Journal of Object-Oriented Programming*, 11(7): 25-35, SIGS Publications, New York.

1998 G. M. Weiss and H. Hirsh. Learning to Predict Rare Events in Categorical Time-Series Data. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, CA, 359-363.

1999 G. M. Weiss. Timeweaver: A Genetic Algorithm for Identifying Predictive Patterns in Sequences of Events. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Morgan Kaufmann, San Francisco, CA, 718-725.

2000 G. M. Weiss and H. Hirsh. A Quantitative Study of Small Disjuncts. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*. AAAI Press, Menlo Park, CA, 665-670.

2002 G. M. Weiss. Predicting Telecommunication Equipment Failures from Sequences of Network Alarms. In W. Kloesgen and J. Zytkow (eds.), *Handbook of Data Mining and Knowledge Discovery*, Oxford University Press, 891-896.

2003 G. M. Weiss and F. Provost. Learning when Training Data are Costly: The Effect of Class Distribution on Tree Induction. *Journal of Artificial Intelligence Research*, to appear.