UNIVERSITY OF CALIFORNIA
IRVINE

A Unified Approach to Concept Learning

DISSERTATION
submitted in partial satisfaction of the requirements for the degree of
DOCTOR OF PHILOSOPHY
in Information and Computer Science
by

Pedro Morais Delgado Domingos

Dissertation Committee:

Professor Dennis F. Kibler, Chair
Professor Michael J. Pazzani
Professor Padhraic Smyth
Professor J. Ross Quinlan

1997

The dissertation of Pedro Morais Delgado Domingos is approved,
and is acceptable in quality and form
for publication on microfilm:

_____

_____

_____

_____

Committee Chair

University of California, Irvine

1997

For my Mother

Contents

List of Figures

List of Tables

Acknowledgments

My thanks go out to all those who directly or indirectly helped me carry out this research. Dennis Kibler was the ideal advisor, giving me both freedom and good advice; he taught me the importance of careful empirical evaluation. Mike Pazzani encouraged me to publish, and taught me by example as well as words how to be a good machine learning researcher. Padhraic Smyth rounded out my background in pattern recognition, statistics and data mining. Ross Quinlan was a source of inspiration, and of more examples of good machine learning research.

Thanks to Rick Lathrop, John King, Dan Hirschberg, Sandy Irani, David Eppstein, David Aha, Lluis Vila, and to my fellow graduate students at UCI, especially Kamal Ali, Cliff Brunk and Chris Merz, for sundry help and fun discussions.

Thanks to the ICS staff, and in particular the systems support group, for providing a smoothly running environment to work in. Thanks to Susan Moore and Theresa Klonecky, and to the people at IIE, CCLA and UCI's International Services, for all their help.

Thanks to my family back home, and to everyone else there, particularly at IST, for their support. Thanks to Gail for everything, and then some. Thanks also to Isabel, Arthur, Kara, Shawne, Alisa, Len, Paul, Sue, Pei, Brian, Carmine, Min, Beki, Shih-Ta, Rui, Mercedes, Gillian, Kathy, Thomas, and the Clarion crowd.

Last but not least, my work at UCI was made possible by Fulbright, IST, CIENCIA, PRAXIS XXI, UC Regents and NATO scholarships.

Curriculum Vitae

Pedro Morais Delgado Domingos

EDUCATION

1997: Ph.D. in Information and Computer Science, University of California, Irvine. Title of dissertation: *A Unified Approach to Concept Learning.* Advisor: Dennis Kibler.

1994: Master of Science in Information and Computer Science, University of California, Irvine. Coursework GPA: 4.0/4.0.

1992: Master of Science in Electrical Engineering and Computer Science, specialization in Computers, Instituto Superior Técnico, Technical University of Lisbon, Portugal. Thesis title: *Competitive Recall: A Memory Model for Real-Time Reasoning.* Advisor: Ernesto Morgado. Coursework GPA: 5.0/5.0.

1988: *Licenciatura* (a 5-year degree) in Electrical Engineering and Computer Science, specialization in Systems and Computers, Instituto Superior Técnico, Technical University of Lisbon, Portugal. Final GPA: 17.2/20.

SCHOLARSHIPS, HONORS AND AWARDS

1997: Recognized as an outstanding reviewer by the program committee of the Fifteenth International Joint Conference on Artificial Intelligence (one of less than 10% of the reviewers).

1997: Invited talk at AT&T Laboratories (Murray Hill, NJ).

1996–97: NATO scholarship.

1992–97: Fulbright Scholarship. Awarded by the United States to 20 out of approximately 3000 candidates in Portugal.

1996: Selected for the SIGART/AAAI Doctoral Consortium.

1996: University of California Regents' Dissertation Fellowship. Awarded to approximately 30 students campuswide.

1996: Invited talk at the University of California, San Diego.

1996: Invited talk at Daimler-Benz Research Center (Ulm, Germany).

1992–96: Scholarship from JNICT, Portugal's national scientific and technological research agency.

1995: Invited talk at the Naval Research Laboratory (Washington, DC).

1995: Invited talk at the University of Porto (Portugal).

1995: Two papers nominated for the C.V. Ramamoorthy Best Paper Award, Seventh IEEE International Conference on Tools with Artificial Intelligence.

1990: Honorable mention in the Descartes Award, given annually to a Portuguese civil servant for original and innovative work in information technology.

1989: Winner of the IEEE Region 8 (Europe, Africa and Middle East) Student Paper Contest.

PROFESSIONAL EXPERIENCE

1994: Consultant for the Irvine Research Corporation.

1987–1992: Teaching and research assistant at Instituto Superior Técnico, Lisbon, Portugal. Taught courses in probability and statistics, applied mathematics (instructor), introduction to computer science, and artificial intelligence.

1990–92: Published a regular column on the future of music technology in the Portuguese magazine *Music, Instruments and Technology*.

1989–90: Developed an AI-based system for personnel selection and job assignment at the Portuguese Army's Center for Psychotechnical Studies.

1986–89: Intern and then researcher at INESC – Institute for Systems and Computer Engineering, Lisbon, Portugal, first in the Digital Signal Processing and Speech Recognition Project, and then in the Computer Graphics Project. Designed a software simulator for the Texas Instruments TMS-320 series of digital signal processors. Created and developed with two coworkers a 3D computer animation system for CAD and simulation, using the C language and UNIX development tools.

1987–88: Teacher of continuing education courses in digital electronics, telecommunications, and introduction to microcomputing.

COMMUNITY SERVICE

1997: Chair of the session on knowledge discovery in databases at AAAI-97, the Fourteenth National Conference on Artificial Intelligence.

1997: Program committee member for AAAI-97, the Fourteenth National Conference on Artificial Intelligence.

1997: Reviewer for IJCAI-97, the Fifteenth International Joint Conference on Artificial Intelligence.

1993–97: Reviewer for *Machine Learning*.

1996: Program committee member for the Workshop on Learning in Context-Sensitive Domains, at the Thirteenth International Conference on Machine Learning, Bari, Italy.

1995: Reviewer for the *Artificial Intelligence Review*.

1990: Founder of the Computer Division of the IST Student Union.

PROFESSIONAL MEMBERSHIPS

Association for Computing Machinery.
ACM Special Interest Group on Artificial Intelligence.
American Association for Artificial Intelligence.
Institute of Electrical and Electronics Engineers.
IEEE Computer Society.
Portuguese Association for Artificial Intelligence.
Portuguese Association for Pattern Recognition.
Portuguese Informatics Association.

BOOK CHAPTERS

"GEAR - A 3D Computer Animation System for CAD and Simulation," with Rui Casteleiro and Pedro Diniz, *IEEE Student Papers*, 1989.

JOURNAL ARTICLES

"On the Optimality of the Simple Bayesian Classifier under Zero-One Loss," with Michael Pazzani, *Machine Learning*, accepted for publication.

"Context-Sensitive Feature Selection for Lazy Learners", *Artificial Intelligence Review*, vol. 11, 1997.

"Unifying Instance-Based and Rule-Based Induction," *Machine Learning*, vol. 24, 1996.

"Two-Way Induction", *International Journal on Artificial Intelligence Tools*, vol. 5, 1996.

"GEAR - A 3D Computer Animation System for CAD and Simulation," with Rui Casteleiro and Pedro Diniz, *Computers and Graphics*, accepted for publication.

"Intelligent Traffic Control for AGV Networks" (in Portuguese), with Paulo Oliveira, *Técnica*, 1992.

"Computer Animation with the GEAR System" (in Portuguese), with Rui Casteleiro and Pedro Diniz, *Ingenium – Journal of the Portuguese Engineering Society*, 1989.

CONFERENCE PUBLICATIONS

"Why Does Bagging Work? A Bayesian Account and its Implications," in *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, Newport Beach, CA, 1997.

"Learning Multiple Models Without Sacrificing Comprehensibility," in *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, Providence, RI, 1997.

"A Comparison of Model Averaging Methods in Foreign Exchange Prediction," in *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, Providence, RI, 1997.

"Knowledge Acquisition from Examples Via Multiple Models," *Proceedings of the Fourteenth International Conference on Machine Learning*, Nashville, TN, 1997.

"Bayesian Model Averaging in Rule Induction," in *Preliminary Papers of the Sixth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL, 1997.

"Towards a Unified Approach to Concept Learning", in *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, OR, 1996.

"Fast Discovery of Simple Rules," in *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, OR, 1996.

"Multistrategy Learning: A Case Study," in *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, OR, 1996.

"Simple Bayesian Classifiers Do Not Assume Independence," with Michael Pazzani, in *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, OR, 1996.

"Using Partitioning to Speed Up Specific-to-General Rule Induction," in *Proceedings of the AAAI-96 Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms*, Portland, OR, 1996.

"Linear-Time Rule Induction," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Portland, OR, 1996.

"Efficient Specific-to-General Rule Induction," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Portland, OR, 1996.

"Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier," with Michael Pazzani, in *Proceedings of the Thirteenth International Conference on Machine Learning*, Bari, Italy, 1996.

"Exploiting Context in Feature Selection," in *Proceedings of the ICML-96 Workshop on Learning in Context-Sensitive Domains*, Bari, Italy, 1996.

"From Instances to Rules: A Comparison of Biases," in *Proceedings of the Third International Workshop on Multistrategy Learning*, Harpers Ferry, WV, 1996.

"Two-Way Induction," in *Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence*, Herndon, VA, 1995.

"Progressive Rules: A Method for Representing and Using Real-Time Knowledge," with Ernesto Morgado, in *Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence*, Herndon, VA, 1995.

"Rule Induction and Instance-Based Learning: A Unified Approach," in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Montréal, Canada, 1995.

"The RISE System: Conquering Without Separating," in *Proceedings of the Sixth IEEE International Conference on Tools with Artificial Intelligence*, New Orleans, LA, 1994.

"Competitive Recall: A Model for Real-Time Reasoning," with Ernesto Morgado, in *Proceedings of the Fourth Iberoamerican Congress on Artificial Intelligence*, Caracas, Venezuela, 1994.

TECHNICAL REPORTS AND UNREFEREED PUBLICATIONS

"Cases or Rules? The Case for Unification," in *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society*, San Diego, CA, 1996.

"Playing Go by Search-Embedded Pattern Recognition," in *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society*, San Diego, CA, 1996.

"The RISE 2.0 System: A Case Study in Multistrategy Learning," Technical Report 95-2, Department of Information and Computer Science, University of California, Irvine, CA, 1995.

"Design and Evaluation of the RISE 1.0 Learning System," Technical Report 94-34, Department of Information and Computer Science, University of California, Irvine, CA, 1994.

"Design and Development of the GEAR Computer Animation System" (in Portuguese), with Rui Casteleiro and Pedro Diniz, in *Second Portuguese Workshop on Computer Graphics*, Porto, Portugal, 1989.

"Towards Artificial Realities" (in Portuguese), *Expresso*, supplement on the Second National Conference on Computer-Aided Design, Planning and Production, 1989.

"Introduction to the GEAR Computer Animation System," with Rui Casteleiro and Pedro Diniz, in *First Luso-German Workshop on Computer Graphics*, Lisbon, Portugal, 1988.

Abstract of the Dissertation

A Unified Approach to Concept Learning

by

Pedro Morais Delgado Domingos

Doctor of Philosophy in Information and Computer Science

University of California, Irvine, 1997

Professor Dennis F. Kibler, Chair

This dissertation proposes a unification of two leading approaches to concept learning: rule induction and instance-based learning.

Current rule induction algorithms based on the "separate and conquer" paradigm suffer from the fragmentation of the training set produced as induction progresses, and from high error rates in rules covering few examples (the "small disjuncts problem"). Current instance-based learners are unable to select different attributes in different regions of the instance space. The limitations of either approach can be addressed by bringing in elements of the other.

In this dissertation, the two paradigms are unified by noting the relationship between the representations they use, and introducing a new algorithm to learn concept descriptions in the unified representation. Instances and rules are unified syntactically by viewing instances as maximally specific rules, and semantically by allowing rules to match examples approximately. The RISE algorithm learns rules by gradually generalizing instances until no improvement in accuracy is obtained. Theoretical analysis shows this approach to be efficient and asymptotically optimal. An extensive empirical study using benchmark datasets shows that RISE consistently succeeds in improving on the predictive accuracy of its parent paradigms, and also on the accuracy of state-of-the-art decision tree learners. Lesion studies verify that each of RISE's components is essential to its performance. Studies in carefully controlled artificial domains show that RISE's advantage relative to other rule induction algorithms is at least partly due to its ability to reduce the fragmentation and small disjuncts problems, and that its advantage relative to other instance-based learners is at least partly due to its ability to select different attributes in different regions of the instance space.

The application of RISE to large databases is made possible through the use of sampling techniques, most notably partitioning, which can sometimes simultaneously reduce running time and improve accuracy. Finally, a data mining algorithm based on RISE's "conquering without separating" strategy is introduced, and shown to have linear worst-case running time in all the relevant parameters, while achieving accuracies at the level of more expensive state-of-the art systems, producing much simpler output, and being highly robust with respect to noise.

Chapter 1
Introduction

1.1 Empirical Multistrategy Learning

Induction is one of the central problems in artificial intelligence, psychology, epistemology, pattern recognition, and statistics. It can be defined in one form as the explicit or implicit creation of general concept or class descriptions from examples. In a typical induction problem, a training set of preclassified examples is given, where each example is described by a vector of attributes or in some other language, and the goal is to form a description that can be used to classify previously unseen examples with high accuracy. Other factors, like speed and comprehensibility of the description, can also be important. The last decade has witnessed renewed interest in this area, partly due to its relevance to the "knowledge acquisition bottleneck" problem: the costliest component in the creation and deployment of an expert system is the construction of the knowledge base, and if this construction can be partly automated by the use of induction techniques, the bottleneck will be greatly reduced. A newer, and even more powerful, source of impetus has been the explosion of information available in computerized form that has taken place in recent years. As large databases have become the norm in many fields (including astronomy, molecular biology, finance, marketing, health care, and many others), the use of induction techniques to discover patterns in them has become a potentially very productive enterprise. Many companies are staking a large part of their future on these "data mining" applications, and looking to the research community for solutions to the fundamental problems they encounter (Fayyad & Uthurusamy, 1995; Simoudis, Han & Fayyad, 1996).

Given the long history and recent growth of the field, it is not surprising that several mature approaches to induction are now available to the practitioner. These include induction of decision trees (Quinlan, 1993a), rule induction (Michalski, 1983), instance-based learning (Aha, Kibler & Albert, 1991), Bayesian classification (Buntine, 1990), neural networks (Rumelhart, Hinton & Williams, 1986), genetic algorithms (Booker, Goldberg & Holland, 1989), and statistical methods (Agresti, 1990). Empirical comparison of these different approaches and their variants in a wide range of application domains has shown that each performs best in some, but not all, domains. This has been termed the *selective superiority* problem (Brodley, 1995), and presents a dilemma to the knowledge engineer approaching a

new task: which induction paradigm should be used? One solution is to try each one in turn, and use cross-validation to choose the one that appears to perform best (Schaffer, 1994a). This is a long and tedious process, especially considering the large number of algorithms and systems now available, and the fact that each typically has options and parameters that themselves need to be fine-tuned by cross-validation or a similar method before the system can be said to be doing its "best."

Another approach, known as multistrategy learning (Michalski & Tecuci, 1994), attempts to combine two or more different paradigms in a single algorithm. Most research in this area has been concerned with combining empirical (i.e., purely inductive) approaches with analytical ones (e.g., (Pazzani & Kibler, 1992; Ourston & Mooney, 1994; Towell & Shavlik, 1994); see also (Michalski & Tecuci, 1993)). The expression "empirical multistrategy learning" will therefore be used to distinguish the case where all the components are empirical (e.g., (Brodley, 1995; Zhang, 1990; Wolpert, 1992; Breiman, 1996d)). Ideally, an empirical multistrategy learning algorithm would always perform as well as the best of its "parents," obviating the need to try each one and simplifying the knowledge acquisition task. Even more ambitiously, there is hope that this combination of paradigms might produce synergistic effects (e.g., by allowing different types of frontiers between classes in different regions of the example space), leading to levels of accuracy that neither atomic approach by itself would be able to achieve. Indeed, in many application domains the classification accuracy of even the best methods is far below 100%, and the question of whether it can be improved, and if so how, is an open and important one.[1]

Unfortunately, this approach has often been only moderately successful. The resulting algorithms are prone to be cumbersome, and often achieve accuracies that lie between those of their parents, instead of matching the highest.[2] Here a theoretical question arises. It is well known that no induction algorithm can be the best in all possible domains; each algorithm contains an explicit or implicit bias (Mitchell, 1980) that leads it to prefer certain generalizations over others, and it will be successful only insofar as this bias matches the characteristics of the application domain. Further, recent results show that performance over the set of all possible domains is subject to a "conservation law" (Schaffer, 1994b) or "no free lunch theorem" (Wolpert, 1996): if one algorithm is better than another in some domains, then there are necessarily other domains in which this relationship is reversed. The average accuracy of an algorithm over all domains is a constant,

[1] One part of answering this question is determining the maximum accuracy achievable by any learner in the application domain (or, conversely, the minimum error rate, known as the *Bayes rate* (Duda & Hart, 1973)). If existing learners do not reach this level, new approaches are needed. Although this problem has received substantial attention (e.g., (Dasarathy, 1991; Cortes, 1995; Tumer & Ghosh, 1996)), no generally reliable method has so far been demonstrated.

[2] However, if multiple models of the same type are combined (e.g., multiple decision trees), as opposed to models from different paradigms (e.g., decision trees and instance-based learning), significant improvements in accuracy often result (Breiman, 1996a; Freund & Schapire, 1996)).

independent of the algorithm. Should we conclude, then, that empirical multi-strategy learning is doomed to failure?

Not necessarily. A distinction should be made between all the mathematically possible domains, which are simply a product of the representation languages used, and the domains that occur in the real world, and are therefore the ones of primary interest (Rao, Gordon & Spears, 1995). Without doubt there are many domains in the former set that are not in the latter, and average accuracy in the real-world domains can be increased at the expense of accuracy in the domains that never occur in practice. Indeed, achieving this is, in a nutshell, the goal of inductive learning research. It is still true that some algorithms will match certain classes of naturally-occurring domains better than other algorithms, and so achieve higher accuracy than them, and that this may be reversed in other real-world domains; but this does not preclude an improved algorithm from being as accurate as the best in each of the domain classes.

Two induction paradigms that appear to have largely complementary strengths and weaknesses are rule induction and instance-based learning (IBL). IBL algorithms are able to induce complex frontiers from relatively few examples and are naturally suited to numeric domains, but can be very sensitive to irrelevant attributes. Conversely, rule induction algorithms perform well at finding simple axis-parallel frontiers, are well suited to symbolic domains, and can often dispose easily of irrelevant attributes; but they can have difficulty with non-axis-parallel frontiers, and suffer from the *fragmentation* problem (i.e., the available data dwindles as induction progresses (Pagallo & Haussler, 1990)) and the *small disjuncts* problem (i.e., rules covering few training examples have a high error rate (Holte, Acker & Porter, 1989)). Of course, the two paradigms also share a number of characteristics, most notably the assumption that the example space contains large continuous regions of constant class membership—the similarity hypothesis (Rendell, 1986).

Instances and rules also form the basis of two alternative approaches to reasoning: case-based reasoning (Kolodner, 1993) and the rule-based reasoning more often found in expert systems. In recent years, case-based reasoning has gained popularity as an alternative to rule systems, but its proponents recognize that there is a wide spectrum from specific cases to the very general rules typically used (Riesbeck & Schank, 1989), and it deserves to be further explored.

This dissertation unifies rule induction and instance-based (or case-based) learning into a single, simple and coherent learning model. This unification rests on two key observations. One is that an instance or case can be regarded as a maximally specific rule (i.e., a rule whose preconditions are satisfied by exactly one instance). Therefore, no syntactic distinction need be made between the two. The second observation is that rules can be matched approximately, as instances are in an instance-based classifier (i.e., a rule can match an example if it is the closest one to it according to some similarity-computing procedure, even if the example does not logically satisfy all of the rule's preconditions). A rule's extension, like

an instance's, then becomes the set of examples that it is the most similar rule to, and thus there is also no necessary semantic distinction between a rule and an instance.

This idea is given a practical, computationally efficient realization in the RISE algorithm. RISE starts with a rule base that is simply the training set itself, and gradually generalizes each rule to cover neighboring cases, as long as this does not decrease the rule base's accuracy on the known cases. If no generalizations are performed, RISE acts as a pure instance-based learner. If all cases are generalized and the resulting set of rules covers all regions of the instance space that have nonzero probability, it acts as a pure rule inducer. More generally, it will produce rules along a wide spectrum of generality; sometimes a rule that is logically satisfied by the target case will be applied, and in other cases an approximate match will be used. This unified model is more elegant and parsimonious than the subprocedure-style combination often found in multistrategy learning systems. Experiments with a large number of benchmark classification problems, reported in this dissertation, have also shown it to consistently outperform either of the component approaches alone, and lesion studies and experiments in artificial domains have confirmed that its power derives from its ability to simultaneously harness the strengths of both components.

It is often the case in computer science that practice runs far ahead of theory. Solutions are found, and systems are built, that perform useful functions but are not amenable to clean theoretical analysis. This does not mean that rigor should be abandoned, however: it means that it should be sought in the systematic empirical evaluation and investigation of the approaches proposed. This dissertation is mainly an instance of this type of work.

1.2 Overview of this Dissertation

This dissertation is structured as follows:

- Chapter 2 introduces the basic notions and terminology of inductive learning, describes the two paradigms that this dissertation unifies—rule induction and instance-based learning—and briefly reviews other approaches to induction.
- Chapter 3 presents the RISE algorithm—its representation scheme, search strategy, and classification procedure. It then derives an upper bound for its time complexity, and bounds for its large-sample error rate. Finally, RISE is compared with previous work.
- Chapter 4 describes the empirical evaluation of RISE that was carried out: the application domains used, the experimental methodology followed, the results obtained in terms of accuracy, learning time, and output simplicity, and their interpretation, aided by lesion (ablation) studies and instrumentation of the algorithm.

- Chapter 5 considers RISE as a rule induction algorithm, compares its properties to those of previous approaches through studies in carefully designed artificial domains, and describes and evaluates its combination with other forms of rule induction.
- Chapter 6 considers RISE as an instance-based learner, derives from it a novel solution to the crucial problem of attribute selection, and compares this solution to classic methods through studies in artificial and benchmark domains.
- Chapter 7 focuses on the application of some of the ideas developed in this dissertation to data mining problems, where speed of learning is a major consideration. Two methods for speeding up RISE—windowing and partitioning—are described and empirically compared. CWS, an algorithm that embodies RISE's approach to rule induction in a form that guarantees linear learning time, is presented and evaluated in terms of speed, accuracy, and comprehensibility of the results produced.
- Chapter 8 reviews the main contributions of this dissertation, and outlines directions for future research.

Chapter 2
Approaches to Concept Learning

2.1 Overview

This chapter defines the concept learning problem, presents a framework for viewing approaches to it, describes in some detail the two paradigms that this dissertation unifies—rule induction and instance-based learning—and briefly reviews other major approaches.

2.2 The Supervised Concept Learning Problem

Strictly speaking, any form of inference in which the conclusions are not deductively implied by the premises can be thought of as induction. This dissertation deals with one specific form of inductive inference, often referred to as *supervised concept learning* or *classification learning*. Although many types of induction exist, supervised concept learning can be considered a key one. Along with regression and probability estimation, it is one of the most studied, and arguably one of those with the greatest practical relevance. Solutions to it are often building blocks of solutions to other induction and learning problems. The usefulness of progress in this area is potentially multiplied many times by its effects on other areas, both within machine learning and in applications.

To *learn* a concept is to infer its general definition from a number of specific examples of it. This definition may be either explicitly formulated or left implicit, but either way it assigns each possible example to the concept or not. Thus a concept can be formally regarded as a function from the set of all possible examples to the Boolean set {True, False} or {1, 0}, or equivalently {Concept, Not_Concept}.[1]

The set of all possible examples is called the *instance space*. Examples can be described in a variety of languages. Most frequently, they are described by means of vectors of attributes. An attribute is simply a variable, which can typically

[1] When an example's description does not uniquely determine whether it belongs to the concept, the concept is *probabilistic*, and can be regarded as a function from the set of all possible examples to the [0, 1] interval. The function's output is the probability that the example belongs to the concept. This important case is discussed in more detail below.

be *symbolic* or *numeric*. A symbolic attribute can take only a finite number of values, among which there is no ordering relationship. For example, the attribute "Body_Covering" with values {Hair, Scales, Feathers, None} is a nominal one. Symbolic attributes are often also referred to as *nominal* or *categorical*. A symbolic attribute which takes values only in the set {True, False} is said to be *Boolean*. By extension, any symbolic attribute that can take only two values is often also called Boolean. A numeric attribute is one that takes values in the set of real numbers, or a subset of it. If the attribute can take non-integer values, it is also referred to as *continuous* or *linear*; if its values are ordered, but the difference between successive values is not well-defined, it is referred to as *ordinal*.

An attribute to which a specific value has been assigned is often called a *feature*. For example, "Body_Covering = Hair" is a feature of mammals. Confusingly, the word "feature" is sometimes also used interchangeably with "attribute." The intended meaning of the word should be inferable from the context.

Besides being definable as a function, a concept can also be formally regarded as a subset of the instance space. For practical purposes, this definition is equivalent to the previous one. Finally, a concept can be regarded as a logical predicate; examples that satisfy the predicate (i.e., for which the predicate is true) are members of the concept.

The set of examples from which the concept is to be induced is called the *training set* or *sample*. If the examples in the training set are labeled with their concept membership, the problem is one of *supervised* concept learning. If the concept membership of the training examples is not given, and the goal is to formulate new concept descriptions that have some desired properties, the problem is one of *unsupervised* concept learning, also known as *clustering* or *concept formation* (Everitt, 1980; Fisher, 1987). The subject of this dissertation is supervised concept learning; for brevity, from here on this will be referred to simply as concept learning.

In supervised learning, the concept to be learned is called the *target concept*. Examples of the concept in the training set are called *positive* examples. Examples in the training set that do not belong to the target concept are called *negative* examples. If the examples in the training set are presented and used all at once, learning is said to be *batch* or *offline*. If the examples are presented one at a time, and the concept definition evolves over time as successive examples are incorporated, learning is said to be *incremental* or *online*. This dissertation concentrates on batch learning.

When several concepts are mutually exclusive (i.e., each example belongs to at most one concept) and are learned concurrently, these concepts are also known as *classes*,[2] and supervised concept learning can then be referred to as *classification learning* (Duda & Hart, 1973). In this case the goal is to learn a

[2]In the most frequent formulation, the set of classes is also exhaustive (i.e., each example belongs to at least one class); however, a "Reject" or "None of the above" option is sometimes

function from the instance space to the set of classes. Conversely, the problem of learning a single isolated concept can be viewed as a two-class classification problem, where the two classes are the concept and its negation. In this framework, concept and classification learning are equivalent; the two expressions will be used interchangeably in this dissertation. However, in some areas (e.g., linguistics and cognitive psychology) more significance may be attached to the notion of a concept than the simple one of a subset of instances or a function from instance space to the set {True, False}, and similarly for the notion of a class. In this case the above equivalence may break down.

In the limit, an infinite number of classes may be present. This will be the case if the target class variable is real-valued. In this view, concept learning also encompasses the tasks of regression and probability estimation (Draper & Smith, 1981; Weiss & Indurkhya, 1995; Good, 1965). The induction of such "continuous" concepts will not be addressed here.

A concept learning algorithm can be viewed as having three components: representation, search, and evaluation (Fayyad, Piatetsky-Shapiro & Smyth, 1996). The *representation* component is the formal language in which concepts are described; the output of the learning algorithm is a statement in this language.[3] The *search procedure* is the process by which the learning algorithm finds the concept description in the space of possible descriptions defined by the representation language. In simple cases this process may be a one-time computation not explicitly involving search. The *evaluation* component takes a candidate concept description and returns a measure of its quality. This is used to guide the search, and possibly to decide when to terminate it. Often, different evaluation procedures are used for these two purposes.

Associated with every concept learning algorithm is a corresponding performance component, which uses the model (i.e., the class definitions) produced by the learner to infer the class of each example that is submitted to it. A system that does this is often called a *classifier*. Figure 2.1 shows in diagrammatic form the main components of a concept learning system and their relationships.

The main goal of a classification learning system is typically (but not always) to produce a classifier that will assign previously-unseen examples (i.e., examples not in the training set) to the corresponding classes with high accuracy.[4] The

---

also allowed, whereby the classifier can decline to assign an example to any of the classes, either because it would not make sense to do so, or because the classifier is unable to make the decision with a minimum degree of confidence.

[3]Note that concepts and examples will in general be represented in different languages; some learning algorithms use the same language for both purposes.

[4]Alternatively, the goal can be stated as accurately classifying any examples, whether previously seen or not. However, in most practical applications exact repetitions of previously-seen examples are very unlikely to appear, rendering the two definitions equivalent. Also, accuracy on training examples is a performance measure of doubtful utility, since it can be trivially maximized by simply storing the examples and looking them up as needed; the real test of a classifier is how well it generalizes to new examples.

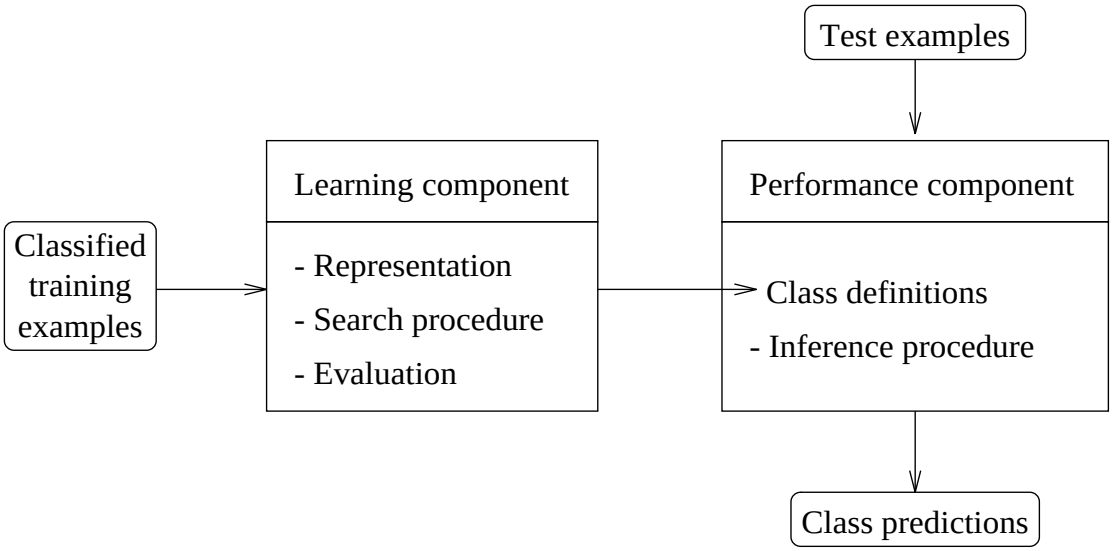

Figure 2.1. Components of a concept learning system.

accuracy of a classifier is defined as the probability that it will correctly classify a new, unlabeled example. This accuracy can be estimated by presenting the classifier with unlabeled examples from a *test set*. For example, if the classifier is presented with 100 test examples and correctly classifies 75, its accuracy can be estimated as 75%.

Ideally, given a complete description of an example (i.e., given the values of all its attributes), its class should be unambiguously determined. In practical learning tasks, however, the available attributes will often not contain all the information necessary to do this. The training set may contain examples with the same attribute values, but different classes. Also, examples may appear with erroneous class values, or with erroneous attribute values, or both. These errors may stem from a wide diversity of sources, including limitations of measuring instruments, and human error while typing examples into a computer. All these phenomena are referred to collectively as *noise*, and limit the achievable accuracy in an induction problem. A learning system's degree of robustness with respect to noise is one of its most important characteristics.

It also occurs often in practice that the values of certain attributes for certain examples are simply not available. These are called *missing values*, and again a practical induction system must be able to handle them.

A great many important problems can be viewed as concept learning. Many examples are given in Section 4.2, and used as applications in this dissertation. They come from the fields of medical diagnosis, molecular biology, finance, social science, engineering and others.

Induction is only one type of learning. In general, any change in a system that causes its performance to improve can be considered learning (Simon, 1983).

Other types of learning are described in (Mitchell, Keller & Kedar-Cabelli, 1986; Minton, 1990; Sutton, 1992; Michalski, Carbonell & Mitchell, 1983; Bellman, 1961; Widrow & Stearns, 1985) and elsewhere. Inductive and non-inductive aspects can be combined in a single learning algorithm (e.g, (Pazzani & Kibler, 1992)).

2.3 Rule Induction

Rule induction algorithms are characterized by representing the target concept as a set of "IF ... THEN ..." rules (Michalski, 1983; Michalski, Mozetic, Hong & Lavrac, 1986; Rivest, 1987; Weiss, Galen & Tadepalli, 1987; Clark & Niblett, 1989; Pagallo & Haussler, 1990; Clearwater & Provost, 1990; Goodman, Higgins, Miller & Smyth, 1992; Segal & Etzioni, 1994; Cohen, 1995). A *rule* is composed of a *consequent* and an *antecedent part* or *body*. The consequent or "THEN" part is the predicted class. The body or "IF" part is a conjunction of *antecedents*. Each antecedent is typically a condition involving the value(s) of a single attribute. For symbolic attributes, this condition is most often a simple equality test; in some algorithms, negation and disjunction of values are possible. For numeric attributes, the condition is typically inclusion in a one-sided interval. An example of a rule is "IF Organic = False AND Mobile = True AND IQ > 75 THEN Robot" where the attributes in the first two conditions are Boolean and the attribute in the third condition is numeric. The previous definitions are valid for "flat" propositional rule induction, on which this dissertation concentrates. "Flat" refers to the fact that no rule chaining is performed; all rules make a direct class prediction, as opposed to possibly having intermediate concepts as consequents, which will in turn be used (directly or indirectly) to predict the target concept. "Propositional" refers to the fact that the expressive power of these rule sets is equivalent to that of propositional logic (or approximately so, depending on the exact formulation). In relational rule induction algorithms (Muggleton & Feng, 1990; Quinlan, 1990), the antecedents and consequent are predicates in first-order logic. In either case, a rule is said to *cover* an example, and conversely the example is said to *satisfy* it, if all the antecedents in the rule are true for the example.

Rule induction algorithms typically employ a set covering or "separate and conquer" search strategy. This strategy derives its name from the fact that it forms a class definition by constructing a rule that covers many positive examples, and few or no negative ones, then "separating out" the newly covered examples and starting again on the remainder. It is summarized in pseudo-code in Table 2.1.[5] The "best" rule in each covering cycle (see table) may also be found by beam search (e.g., (Clark & Niblett, 1989; Clearwater & Provost, 1990)). In this case a list of the b best rule bodies found so far is maintained, instead of a single body. At each step, specialization of each of those bodies with each possible antecedent

[5] Variations of this approach optimized for efficiency will be discussed in Chapter 7.

is attempted, and the best b bodies are selected to continue the search. At the end the best rule body overall is selected.

Another alternative is (near-)exhaustive search, where all (or nearly all) possible rules are attempted, using pruning techniques to avoid unnecessary computation (Weiss, Galen & Tadepalli, 1987; Goodman, Higgins, Miller & Smyth, 1992; Rymon, 1993; Segal & Etzioni, 1994; Webb, 1995). This approach can have high computational cost, and often hurts generalization accuracy instead of improving it. This is essentially due to the fact that, when a very large space of possible rule sets is exhaustively searched, there is a high probability of finding a rule set that is highly accurate on the training data purely by chance. This rule set will be chosen over others that are in fact more accurate outside the training set, leading to poorer results (Quinlan & Cameron-Jones, 1995). Because of its cost and problematic effect on accuracy, exhaustive search is seldom used in practice.

The choice of evaluation heuristic H (see Table 2.1) is of some importance to the performance of a "separate and conquer" algorithm. Given a rule, H should increase with $e_\oplus$, the number of positive examples that satisfy the rule, and decrease with $e_\ominus$, the number of negative examples that satisfy it. The AQ series of algorithms (Michalski, Mozetic, Hong & Lavrac, 1986) uses apparent accuracy (i.e., the accuracy of the rule on the training set):

$$H(e_\oplus, e_\ominus) = \frac{e_\oplus}{e_\oplus + e_\ominus} \tag{2.1}$$

The CN2 system (Clark & Niblett, 1989) originally used the entropy of the rule (Quinlan, 1986). However, the problem with both these measures is that they tend to favor overly specific rules: they attain their maximum value with a rule covering a single example. This can be overcome by use of the Laplace correction (Niblett, 1987; Good, 1965):

$$H(e_\oplus, e_\ominus) = \frac{e_\oplus + 1}{e_\oplus + e_\ominus + c} \tag{2.2}$$

where c is the number of classes. This measure approaches the uncorrected accuracy when the rule has strong statistical support (i.e., when it covers many examples), but approaches $1/c$ (i.e., "maximum ignorance") when it covers few. It is used in recent versions of CN2 (Clark & Boswell, 1991). An alternative approach is to use a measure that takes into account the rule's probability of matching an example ($(e_\oplus + e_\ominus)/e$, where e is the training set size), as in (Goodman, Higgins, Miller & Smyth, 1992). Further rule evaluation measures are described in (Piatetsky-Shapiro, 1991).

Classification of a new example is performed by matching each rule against it, and selecting those it satisfies. If there is only one such rule, its class is assigned to the example. If there are none, the generally adopted solution is to use the

Table 2.1: General structure of "separate and conquer" rule induction algorithms.

Input: ES is the training set.

Procedure Rule_Induction (ES)
Let $RS = \emptyset$.
For each class C
  Let $\oplus = \{E \in ES \mid \text{Class}(E) = C\}$.
  Let $\ominus = \{E \in ES \mid \text{Class}(E) \neq C\}$.
  Repeat
    Let R = Find_Best_Rule (C, $\oplus$, $\ominus$).
    Let $\oplus = \oplus - \{E \in \oplus \mid R \text{ covers } E\}$.
    Let $RS = RS \cup \{R\}$.
  Until $\oplus = \emptyset$ or R = Nil.
Return RS.

Function Find_Best_Rule (C, $\oplus$, $\ominus$)
Let Body = True.
Let R be the rule: Body $\Rightarrow C$.
Repeat
  For each possible antecedent A
    Let B_A = Body $\wedge A$.
    Let $e_\oplus = \#(\{E \in \oplus \mid E \text{ satisfies } B_A\})$.
    Let $e_\ominus = \#(\{E \in \ominus \mid E \text{ satisfies } B_A\})$.
  Let Body = B_A that maximizes some heuristic $H(e_\oplus, e_\ominus)$.
Until no antecedent causes a significant improvement in $H(e_\oplus, e_\ominus)$.
Return R, or Nil if Body = True.

---

so-called "default rule" (i.e., to assign the example to the class that occurs most frequently in the entire training set, or among those examples not covered by any rule). Finally, if more than one rule covers the example, then two strategies are possible. One is to order the rules into a *decision list*, typically in the order in which they were induced, and select only the first rule that fires (Rivest, 1987). This is equivalent to having an "IF ... THEN ... ELSE IF ... THEN ... ELSE ..." statement, with an "ELSE IF ..." branch for every rule after the first, and a final "ELSE ..." for the default. The other strategy is to let the different rules vote, and select the class receiving the highest vote. Recent versions of CN2 attach to each rule the number of examples of each class that it covers, and use these numbers as votes at classification time (Clark & Boswell, 1991). Other voting schemes are possible (e.g., (Michalski, Mozetic, Hong & Lavrac, 1986; Goodman, Higgins, Miller & Smyth, 1992)). The use of unordered rules has been found to generally produce higher accuracy (Clark & Boswell, 1991), and also has the advantage of greater comprehensibility, since in a decision list each rule body is implicitly conjoined with the negations of all those that precede it.

In rule induction algorithms that do not deal with noise, construction of a new rule stops only when all negative examples are excluded. In noise-tolerant ones, a measure of statistical significance may be used to halt growth (as in CN2). Irrelevant attributes tend to produce no significant improvement in the evaluation heuristic, and thus to be excluded. However, attributes that are relevant only in combination with other attributes may also be discarded. Alternatively, a later post-pruning step may be used to remove superfluous antecedents and/or rules (e.g., GROVE (Pagallo & Haussler, 1990)).

The fact that most rule learners only use single-attribute tests in constructing rules means that the decision boundaries they produce will necessarily be parallel to the coordinate axes (i.e., class definitions can only be unions of hyperrectangles), leading to inaccurate definitions when boundaries are non-axis-parallel and only a limited number of examples is available. Another shortcoming of "separate and conquer" rule induction is that it causes a dwindling number of examples to be available as induction progresses, both within each rule and for successive rules. This splintering of the training set may cause later rules, and later antecedents within each rule, to be induced with insufficient statistical support, leading to greater noise sensitivity and missing or incorrect rules/antecedents. This is known as the *fragmentation problem* (Pagallo & Haussler, 1990).

Rule induction algorithms also suffer from the *small disjuncts problem*, first observed by Holte *et al.* (1989): rules covering few training examples (less than five, say) tend to be highly error-prone, but removing them often increases the global error even further. For example, the global accuracy may be 95%, and the accuracy of a given small disjunct 75%, but when this disjunct is removed the larger ones that now classify the uncovered examples have an accuracy of 60% on them, decreasing the global accuracy. Some small disjuncts correspond to rare cases of the concept, and are intrinsically difficult to learn, because only a small

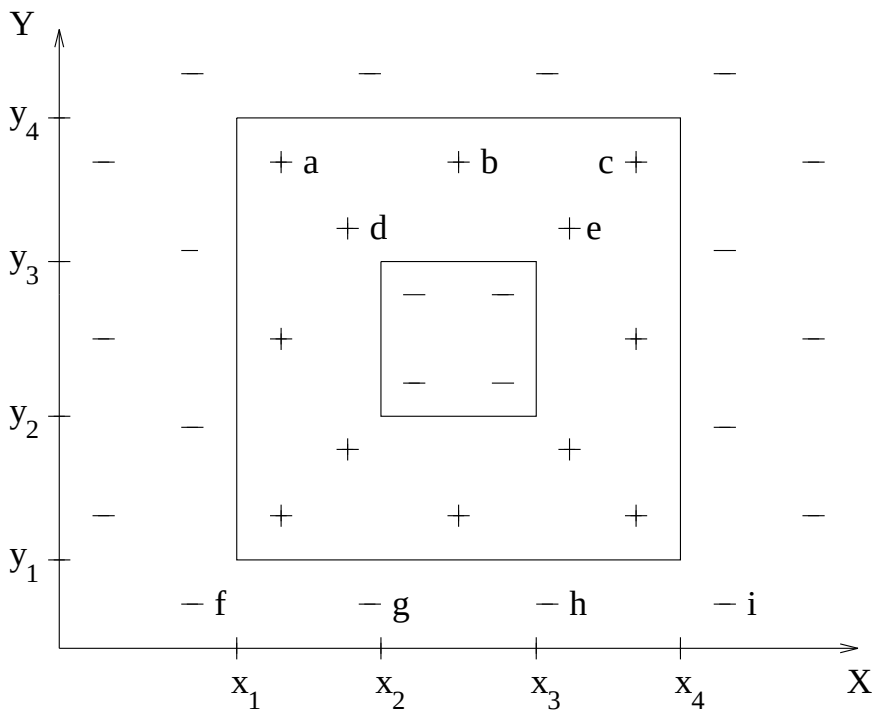

Figure 2.2: A simple concept illustrating the fragmentation and small disjuncts problems.

sample of them is available. However, other small disjuncts may be at least partly a product of the fragmentation problem. This is illustrated by the example in Figure 2.2, where the instance space is the XY plane, the target concept is the square doughnut region between the two closed solid lines, and + and − indicate the positive and negative examples in the training set, respectively. Some of the examples are labeled with letters (a to i). This concept can be represented by the following decision list (where the rules are numbered in the left margin):

1. IF $X < x_1$ THEN −
2. ELSE IF $X > x_4$ THEN −
3. ELSE IF $Y < y_1$ THEN −
4. ELSE IF $Y > y_4$ THEN −
5. ELSE IF $X < x_2$ THEN +
6. ELSE IF $X > x_3$ THEN +
7. ELSE IF $Y > y_3$ THEN +
8. ELSE IF $Y < y_2$ THEN +
9. ELSE −

Rules 1 and 2 are easily induced. However, when inducing rule 3, only examples g and h are available; examples f and i should ideally also be used, but they have been discarded because they were covered by the previous rules. This complicates induction of rule 3. Similar considerations apply to rule 4. The problem is even more acute after rules 5 and 6 have been induced: only example b is

now left to induce rule 7, when all the five examples a to e should be. The chances of the correct rule being induced are small. Rule 8 will be similarly affected. The rules that will actually be induced may appear to be small disjuncts, when in fact they should cover enough examples to make their reliable induction possible. The small disjuncts here do not correspond to rare cases, but are an artifact of the fragmentation problem.

This example, with only two features present (X and Y), understates the seriousness of the problem; in the high-dimensional domains commonly found, the sparseness of training data is much greater, even if training sets are large, and there are correspondingly more opportunities to make the wrong decision.

Rule induction algorithms have several advantages, the most notable one of which is perhaps that, of all representations currently in use in concept learning, rules are arguably the one most easily understood by humans (Michie, Spiegelhalter & Taylor, 1994; Quinlan, 1993a). This is important not only in adding to the usefulness and acceptability of the learner's output, but in improving the loop of interaction between automatic system and human developer that is at the root of most practical applications of machine learning. Thus the comprehensibility of rule sets is an important feature not only in its own right, but also as a means to improved accuracy.

Other advantages of rule induction algorithms include: their ability to efficiently select relevant attributes in high-dimensional instance spaces, and to automatically use different attributes in different regions of the space; when designed to handle noise, their robustness with respect to it when the target concept is simple; and their natural suitability for symbolic domains, and ability to easily mix symbolic and numeric attributes.

2.4 Instance-Based Learning

Instance-based learning[6] is based on the idea of letting the examples themselves form an implicit representation of the target concept (Cover & Hart, 1967; Duda & Hart, 1973; Stanfill & Waltz, 1986; Porter, Bareiss & Holte, 1990; Aha, Kibler & Albert, 1991; Dasarathy, 1991; Salzberg, 1991; Cost & Salzberg, 1993; Ram, 1993; Aha, 1997). In the simplest case, learning is performed by simply storing all the observed examples. A test example is classified by finding the nearest stored example (i.e., the "nearest neighbor") according to some similarity function, and assigning the latter's class to the former. The stored examples used to classify new ones are referred to as *instances*, *cases* or *exemplars*. These, together with the similarity function used, implicitly define a partition of the instance space into

[6]This phrase will be used in this dissertation to refer to a family of learning approaches that also includes, or is also known as, exemplar-based, memory-based, case-based, experience-based, kernel-based, nearest-neighbor, local, and lazy learning.

regions of each class: a point in instance space belongs to the class of the nearest stored instance. The performance of IBL depends critically on the similarity (or, conversely, distance) metric used. In numeric domains (i.e., domains where all the features are real-valued), city-block and Euclidean distance are natural candidates. The component distance $\delta(x_i, x_j)$ between two values x_i and x_j of an attribute is then simply the absolute value of their difference; however, this can lead to attributes with a large spread of values having undue weight in the result, compared to attributes with smaller spreads. This can be avoided by normalizing the distance along each attribute by the attribute's standard deviation (Michie, Spiegelhalter & Taylor, 1994). Another commonly used approach (e.g., (Aha, Kibler & Albert, 1991; Salzberg, 1991)) is to normalize the difference by its largest observed value:

$$\delta(x_i, x_j) = \left| \frac{x_i - x_j}{x_{max} - x_{min}} \right| \tag{2.3}$$

If there are a attributes, the distance between two instances $E_1 = (e_{11}, e_{12}, \ldots, e_{1a}, C_1)$ and $E_2 = (e_{21}, e_{22}, \ldots, e_{2a}, C_2)$ can then be defined as:

$$\Delta(E_1, E_2) = \sum_{i=1}^{a} \delta^s(e_{1i}, e_{2i}) \tag{2.4}$$

with $s = 1$ yielding city-block distance and $s = 2$ the square of Euclidean distance.

Symbolic attributes pose a more difficult problem. Most IBL systems (e.g., (Aha, Kibler & Albert, 1991)) use a simple overlap metric:

$$\delta(x_i, x_j) = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{otherwise} \end{cases} \tag{2.5}$$

This measure is obviously less informative than its numeric counterpart, and, although it is appropriate in some cases, its use can lead to poor performance (Cost & Salzberg, 1993). A more sophisticated alternative consists of considering two symbolic values to be similar if they make similar predictions (i.e., if they correlate similarly with the class feature). This was first proposed by Stanfill and Waltz (1986) as part of their value difference metric (VDM) for a memory-based reasoner. Here we will consider a simplified version of VDM, which defines the distance between two symbolic values as:

$$\delta(x_i, x_j) = SVDM(x_i, x_j) = \sum_{h=1}^{c} |P(C_h|x_i) - P(C_h|x_j)|^q \tag{2.6}$$

where c is the number of classes, C_h is the hth class, and q is a natural-valued parameter ($q = 1, 2, 3, \ldots$). The latter can be determined *ad hoc* or empirically.

Notice that $\delta(x_i, x_j)$ is always 0 if $i = j$. The total distance $\Delta(E_1, E_2)$ is computed as before. Different variants of this metric have been successfully used in pronunciation, molecular biology and other tasks (Stanfill & Waltz, 1986; Cost & Salzberg, 1993; Biberman, 1994).

Instance-based learners are conceptually simple, and yet able to form complex decision boundaries in the instance space even when relatively little information is available. They combine naturally with analogical reasoning, apply easily to numeric domains, and with distance measures such as VDM can also outperform other approaches in symbolic ones. Special cases that may be missed by abstraction-forming approaches can be retained and recognized. Learning is often simple to perform, because it involves mainly storing the examples, possibly with some selection and indexing.

IBL approaches have some shortcomings, however. The memory cost of the class descriptions they produce is typically greater, and they can be harder for a human to understand. Classification can also take longer, even with suitable indexing schemes (e.g., (Friedman, Bentley & Finkel, 1977)). However, the most significant problem for IBL is arguably that posed by *irrelevant* attributes (i.e., attributes that give no information about the target concept, either directly or in combination with others). The contributions of these attributes to the global distance constitute noise as far as the classification task is concerned, and they can swamp out the relevant components. If many such attributes are present in the example descriptions, instance-based learners will be confused by them when they compare examples, resulting in a possibly severe degradation of accuracy. A natural solution to this problem is identifying the irrelevant attributes, and discarding them before storing the examples for future use. Several algorithms have been proposed for this purpose (see (Kittler, 1986) for a survey), of which two of the most widely known are forward sequential selection (FSS) and backward sequential selection (BSS) (Devijver & Kittler, 1982). Many variations of these exist (e.g., (Aha & Bankert, 1994; Moore & Lee, 1994)).[7] Their use can have a large positive impact on accuracy.

The forward sequential selection algorithm (FSS) starts with an empty feature set and repeatedly adds the "best" feature to it until no further improvement is possible, or all features have been included. The backward sequential selection algorithm (BSS) operates similarly, but starts with the full feature set and repeatedly removes the "worst" feature from it. The two algorithms are described in pseudo-code in Figures 2.2 and 2.3. In both cases, the final feature set can be empty and all examples assigned to the default class, if this leads to the highest accuracy.

[7] Another approach is to assign a weight to each feature, with a weight of zero being equivalent to deletion. This is more flexible, but requires more data to be safely applied. Approaches of this type are discussed in Chapter 6.

Table 2.2. The forward sequential selection (FSS) algorithm.

Input: FS is the set of features used to describe examples.

Procedure FSS (FS)

Let $SS = \emptyset$.
Let $BestEval = 0$.
Repeat
  Let $BestF = None$.
  For each feature F in FS and not in SS
    Let $SS' = SS \cup \{F\}$.
    If $Eval(SS') > BestEval$
      Then Let $BestF = F$,
        Let $BestEval = Eval(SS')$.
  If $BestF \neq None$
    Then Let $SS = SS \cup \{BestF\}$.
Until $BestF = None$ or $SS = FS$.
Return SS.

Table 2.3. The backward sequential selection (BSS) algorithm.

Input: FS is the set of features used to describe examples.

Procedure BSS (FS)

Let $SS = FS$.
Let $BestEval = Eval(SS)$.
Repeat
  Let $WorstF = None$.
  For each feature F in SS
    Let $SS' = SS - \{F\}$.
    If $Eval(SS') \geq BestEval$
      Then Let $WorstF = F$,
        Let $BestEval = Eval(SS')$.
  If $WorstF \neq None$
    Then Let $SS = SS - \{WorstF\}$.
Until $WorstF = None$ or $SS = \emptyset$.
Return SS.

The evaluation function $Eval()$ can be a heuristic measure, typically of the quality of the class separation produced by the feature set, or it can be the actual accuracy obtained by applying the classifier using those features. If $Eval()$ is a heuristic measure, the feature selection algorithm acts as a filter, extracting features to be used later by the main algorithm; if it is the actual accuracy, it acts as a wrapper around that algorithm (John, Kohavi & Pfleger, 1994). The wrapper strategy has been found to often yield the best results (Aha & Bankert, 1994), and this is attributable to the fact that its learning bias is that of the classifier itself, avoiding a possible mismatch between the feature selection and classification biases.

A limitation of FSS and BSS, and of all of their many variants, is that they ignore the fact that some attributes may be relevant only in context (i.e., given the values of other attributes). They may discard attributes that are highly relevant in a restricted sector of the instance space because this relevance is swamped by their irrelevance everywhere else. They may retain attributes that are relevant in most of the space, but unnecessarily confuse the classifier in some regions.

Consider, for example, an instance space defined by a set of numeric attributes A, and a class composed of two hyperrectangles, one of which is defined by intervals $a_i \in [v_{i1}, v_{i2}]$ in a subset A_1 of the attributes, and the other by intervals in a subset A_2 disjoint from the first. Current attribute selection algorithms would retain all attributes in A_1 and A_2, because each of those attributes is relevant to identifying examples in one of the hyperrectangles. However, the attributes in A_2 act as noise when identifying examples defined by A_1, and vice-versa. Instead of storing the same set of attributes for all instances, a better algorithm would discard the attributes in A_2 from the stored instances of the first hyperrectangle, and the attributes in A_1 from those of the second one.

As another example, consider Figure 2.3, where the concept to be learned is the rectangle delimited by the solid line, and + and – indicate the positive and negative examples in the training set, as before. The basic one-nearest-neighbor algorithm with Euclidean distance would produce the boundary shown as a dashed line, resulting in a large error. A context-free attribute selection algorithm would retain both attributes, producing the same boundary, or delete one of them, collapsing the plane to a line and resulting in an even greater error. A context-sensitive algorithm, on the other hand, would ignore attribute X when $Y > y_2$ or $Y < y_1$, because in those areas all examples are negative irrespective of the X coordinate, and it would take X into consideration when $y_1 \leq Y \leq y_2$, because here examples are positive if $x_1 \leq X \leq x_2$, and negative otherwise. X is thus relevant or not depending on the context (i.e., on the value of Y), and recognizing this leads to the correct boundary being induced.

Another issue in IBL methods is their sensitivity to noise. Incorrect instances are liable to create a region around them where new examples will also be misclassified. Several methods have been successfully introduced to deal with this

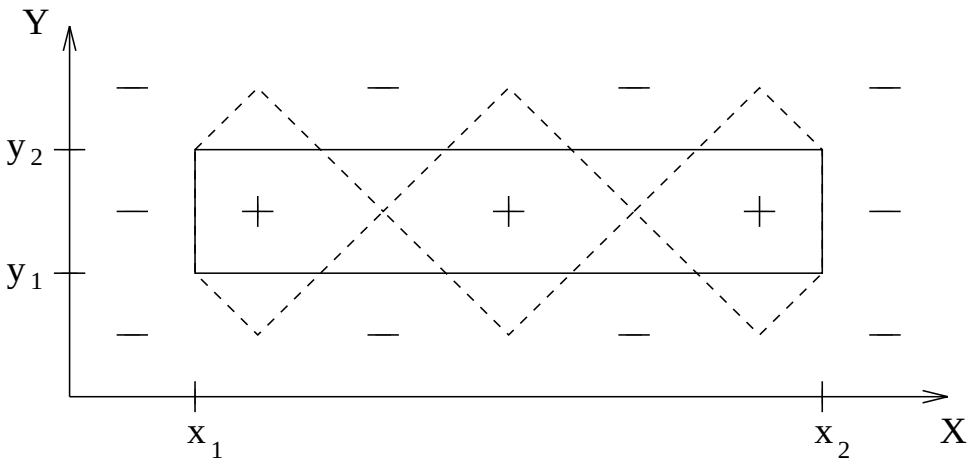

Figure 2.3. A simple concept requiring context-sensitive attribute selection.

problem. IB3 (Aha, Kibler & Albert, 1991) retains only reliable instances, reliability being judged by the instance's classification performance over a "probation period." Cameron-Jones (1992) uses instead a criterion based on the minimum description length principle to decide which instances to retain. PEBLS (Cost & Salzberg, 1993) assigns weights to instances, making their apparent distance to new examples increase with their misclassification rate.

Given two instances of different classes, the frontier between classes induced by them is a hyperplane perpendicular to the line connecting the two instances, and bisecting it. With multiple instances of each class, the frontier will be composed of a number of hyperplanar sections, and can thus become quite complex even when few instances are present (Aha, Kibler & Albert, 1991). The introduction of weights further increases this complexity, turning the hyperplanes into hyperquadrics (Cost & Salzberg, 1993).

Another variation of the basic IBL paradigm consists in using the k nearest neighbors for classification, instead of just the nearest one (Duda & Hart, 1973). The class assigned is then that of the majority of those k neighbors, or the class receiving the most votes, with a neighbor's vote decreasing with its distance from the test example. The best value of k for a given application is difficult to predict *a priori*, and in practice it is typically determined by cross-validation (Michie, Spiegelhalter & Taylor, 1994). If feature selection is also being performed, the combined cost of determining the best k and feature subset can become quite significant. Use of the k nearest neighbors also slows down the classification of new examples.

It is important to note that, even though the stored instances used in classification are syntactically identical to examples, their semantic content (i.e., their extension) is quite different. An example is a single point in the example space, whereas a stored instance represents the entire region that it wins over in the competition with other instances.

2.5 Other Approaches

Besides rule induction and instance-based learning, several other approaches to concept learning exist. This section will briefly review some of the main ones: decision tree induction, neural networks, genetic algorithms, and Bayesian methods.

Decision tree induction is probably the most widely-used approach in machine learning (Quinlan, 1986; Quinlan, 1993a; Breiman, Friedman, Olshen & Stone, 1984). In a decision tree, each node contains a test, in the simplest and most frequent case involving the value of a single attribute. Examples are classified by passing them from the root of the tree down to a leaf, according to the outcomes of the tests along the path. Each leaf contains a class prediction. Figure 2.4 shows a decision tree for the concept of "robot." Each node is labeled with the attribute it tests, and its branches are labeled with the corresponding values.

A decision tree is in effect a sequence of nested "IF ... THEN ..." statements. For example, the tree in Figure 2.4 corresponds to the sequence:

```
IF Organic = True THEN Not_a_Robot
ELSE IF Organic = False THEN
    IF Mobile = False THEN Not_a_Robot
    ELSE IF Mobile = True THEN
        IF IQ ≤ 75 THEN
            IF Functionality = Fixed THEN Not_a_Robot
            ELSE IF Functionality = Programmable THEN Robot
            ELSE IF Functionality = Learnable THEN Robot
        ELSE IF IQ > 75 THEN Robot
```

Decision tree induction is closely related to rule induction. Each path from the root of a decision tree to one of its leaves can be transformed into a rule simply by conjoining the tests along the path to form the antecedent part, and taking the leaf's class prediction as the consequent. The resulting rule set can then be simplified to improve its comprehensibility to a human user, and possibly its accuracy (Quinlan, 1987a; Quinlan, 1987b). The search process most often used to learn decision trees, known as "divide and conquer", is also closely related to the "separate and conquer" method used in rule induction, and has similar difficulties: nodes are added one at a time, using an evaluation heuristic to choose the test, and each node further subdivides the training set into smaller subsets, leading to a lack of data for later induction steps. Thus decision trees will tend to perform well if a few highly relevant attributes exist, but less so if many complex interactions are present.

Neural network methods are based on representing the concept as a network of nonlinear units (Anderson & Rosenfeld, 1988). The most frequently used type of unit, incorporating a sigmoidal nonlinearity, can be seen as a generalization

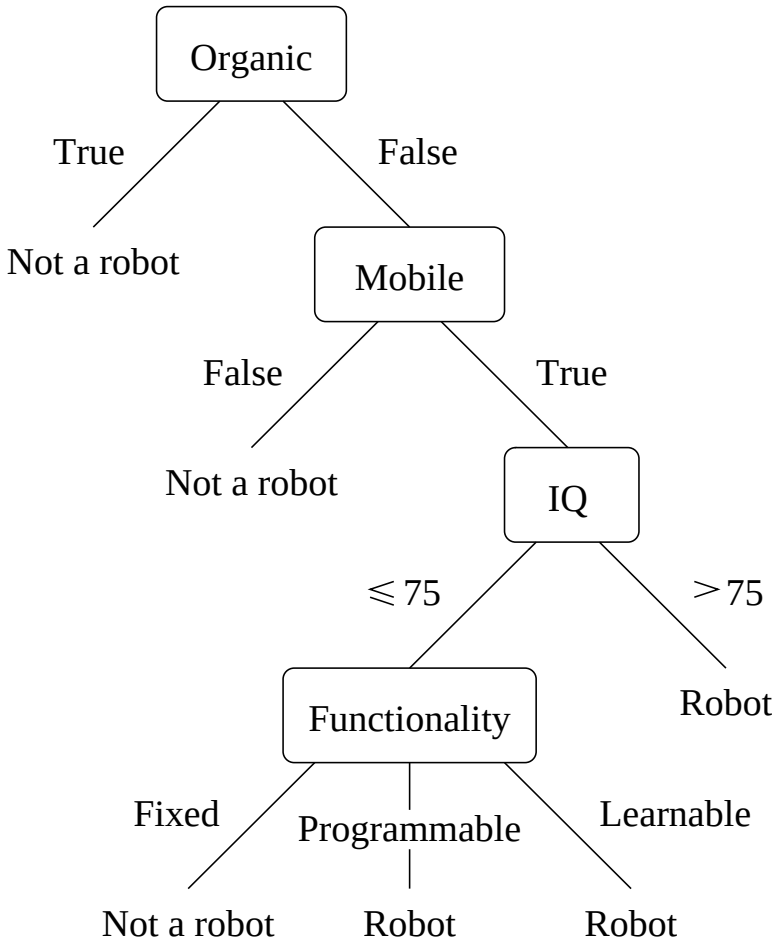

Figure 2.4. A decision tree for the concept of "robot."

of a propositional rule, where numeric weights are assigned to antecedents, and the output is graded, rather than binary (Towell & Shavlik, 1994). Many search methods can be used to learn these networks, of which the most widely applied one is backpropagation (Rumelhart, Hinton & Williams, 1986). This method efficiently propagates values of the evaluation function backward from the output of the network, which then allows the network to be adapted so as to obtain a better evaluation score. Radial basis function (RBF) networks employ units with a Gaussian nonlinearity (Moody & Darken, 1989), and can be seen as a generalization of nearest-neighbor methods with an exponential distance function (Poggio & Girosi, 1990). Neural networks tend to perform better than decision trees when no highly relevant attributes exist, but many weakly relevant ones are present. The cost of this is the often-long time they take to train, both in terms of CPU time, and of manually finding parameter settings that will enable successful learning. Another disadvantage of neural network models is that, being rather opaque, they afford little insight about the underlying domain.

Genetic algorithms are a search method that can be applied to learning many different representations, of which the most frequently used one is probably rule sets (Booker, Goldberg & Holland, 1989). Genetic algorithms maintain a population of classifiers during learning, as opposed to just one, and search by applying random mutations to them, and exchanging parts between pairs of classifiers that obtain high evaluation scores. This endows them with a potentially greater ability to avoid local minima than is possible with the simple greedy search employed in most learners, but can lead to high computational cost, and to higher risks of finding poor classifiers that appear good on the training data by chance.

Bayesian approaches employ probabilistic concept representations, and range from the simple Bayesian classifier (Domingos & Pazzani, 1996) to Bayesian networks, which learn the full joint probability distribution of the attributes and class, as opposed to just the class prediction (Heckerman, 1996). Bayesian networks have the benefit of a clearer semantics than more *ad hoc* methods, and provide a natural platform for combining domain knowledge (in the initial network structure) and empirical learning (of the probabilities, and possibly of new structure). However, inference in Bayesian networks can have a high time complexity, and as tools for classification learning they are not yet as mature or well-tested as other approaches. More generally, as Buntine (1990) notes, the Bayesian paradigm extends beyond any single representation, and forms a framework in which many learning tasks can be usefully studied.

Further details and pointers to the literature on these concept learning paradigms can be found in the references given.

2.6 Summary

This dissertation addresses the problem of supervised concept learning, where the goal is to form a definition of a concept from a set of labeled training examples, such that it will be possible to determine with high accuracy whether new examples are instances of the concept. Two of the leading approaches to this problem are rule induction, where the concept is represented as a set of "IF ... THEN ..." rules, and instance-based learning, where the concept is implicitly represented by a set of examples, together with a similarity measure. Other approaches include induction of decision trees, neural networks, genetic algorithms and Bayesian classification.

Chapter 3
The RISE Algorithm

3.1 Overview

This chapter introduces the RISE algorithm, which unifies rule induction and instance-based learning, and thereby reduces some of the problems of each by bringing in features of the other. RISE's unified representation scheme and classification procedure are presented. RISE's search strategy, used to induce concept descriptions in the unified representation, is then described. Some simple examples of RISE's application are shown, highlighting the improvements it produces over IBL and rule induction. Next, efficiency considerations are addressed, by introducing several optimizations to RISE, and deriving worst-case bounds for its running time. RISE's large-sample error rate is then shown to be at most twice the optimal. Finally, several pieces of related research are contrasted with RISE.

3.2 Representation and Classification

"RISE" stands for "Rule Induction from a Set of Exemplars". This chapter describes version 3.1 of the RISE system; earlier versions are described in (Domingos, 1994a; Domingos, 1994b; Domingos, 1995).

A rule in RISE is composed of a consequent that is the predicted class, and an antecedent part that is a conjunction of conditions, as in other rule induction systems. Each condition involves only one attribute; for symbolic attributes it is an equality test (e.g., $x_1 = \alpha$), and for numeric attributes it is membership in an interval closed on both sides (e.g., $3.5 \leq x_2 \leq 7.7$). In each rule there is at most one condition involving each attribute, and there may be none. An instance is simply a rule in which the consequent is the instance's class, there is exactly one condition per attribute, and all the intervals are degenerate (e.g., $4.1 \leq x_2 \leq 4.1$, i.e., $x_2 = 4.1$). Thus, syntactically, an instance can be regarded as simply a maximally specific rule. In the remainder of this dissertation, the word "rule" is used to refer indiscriminately to instances and to rules of the more general type.

RISE classifies a new example by assigning it the class of the nearest rule in the knowledge base. The distance between a rule and an example is defined

as follows. Let $E = (e_1, e_2, \ldots, e_a, C_E)$ be an example with value e_i for the ith attribute and class C_E. Let $R = (A_1, A_2, \ldots, A_a, C_R)$ be a rule with class C_R and condition A_i on the ith attribute, where A_i can have three forms: if there is no condition on i, A_i is True; if there is a condition on i and i is symbolic, A_i is $e_i = r_i$, where r_i is a symbolic value in attribute i's domain; and if there is a condition on i and i is numeric, A_i is $r_{i,lower} \leq e_i \leq r_{i,upper}$, where $r_{i,lower} \leq e_i$ and $r_{i,upper}$ are numeric values in attribute i's domain. The distance $\Delta(R, E)$ between R and E is then defined as:

$$\Delta(R, E) = \sum_{i=1}^{a} \delta^s(i) \tag{3.1}$$

where s is a natural-valued parameter ($s = 1, 2, 3, \ldots$), and the component distance $\delta(i)$ for the ith attribute is:

$$\delta(i) = \begin{cases} 0 & \text{if } A_i = \text{True} \\ SVDM(r_i, e_i) & \text{if } i \text{ is symbolic and } A_i \neq \text{True} \\ \delta_{num}(i) & \text{if } i \text{ is numeric and } A_i \neq \text{True} \end{cases} \tag{3.2}$$

where in turn $SVDM(r_i, e_i)$ is the simplified value difference metric as defined in Equation 2.6, and:

$$\delta_{num}(i) = \begin{cases} 0 & \text{if } r_{i,lower} \leq e_i \leq r_{i,upper} \\ \frac{e_i - r_{i,upper}}{e_{i,max} - e_{i,min}} & \text{if } e_i > r_{i,upper} \\ \frac{r_{i,lower} - e_i}{e_{i,max} - e_{i,min}} & \text{if } e_i < r_{i,lower} \end{cases} \tag{3.3}$$

$e_{i,max}$ and $e_{i,min}$ being respectively the maximum and minimum values for the attribute found in the training set.

The distance from a missing numeric value to any other is defined as 0. This is a "least commitment" strategy: it lets an example with a missing numeric value be matched by all rules with conditions on the corresponding attribute, and defers the decision on which rule is most appropriate to the conflict resolution procedure (see below). Conversely, an instance/rule with a missing numeric value is effectively treated as having no condition on the attribute, allowing it to match examples regardless of their value for that attribute, and the decision on whether this rule is better than nearby ones with specified values for the attribute is deferred to the evaluation procedure described in the next section. If a symbolic attribute's value is missing, it is assigned the special value "?". This is treated as a legitimate symbolic value, and its SVDM to all other values of the attribute is computed and used. In the context of VDM-type metrics, this is a sensible policy: a missing value is taken to be roughly equivalent to a given possible value if it behaves similarly to it, and distinct from that value if it does not. Other approaches to handling missing values are described in (Quinlan, 1993a; Ripley, 1996).

The question arises of how to choose the winning rule when several are equally near. This is of more importance in RISE than in IBL, because it can frequently occur that several rules cover the example (i.e., are at distance zero from it). This corresponds to the multiple-match case in rule induction systems. RISE selects the rule with the highest Laplace accuracy (Equation 2.2). This means that neither very general nor very specific rules are unduly favored; rather, preference goes to rules with high apparent accuracy as well as strong statistical support. In the event the accuracies are the same, RISE chooses the most frequent class among those represented, and if there is still a draw, the winner is chosen at random. Other policies were also tried, and a comparative evaluation is described in the next chapter.

A rule is said to *cover* an example if all its conditions are true for the example; a rule is said to *win* an example if it is the nearest rule to the example according to the distance metric and conflict resolution policy just described. A rule can cover an example and not win it. The extension of a rule is constituted by all the points in the example space that it wins, whether or not it covers them, and therefore depends not only on the rule itself but on all the other rules. Thus, the semantic content of rules in RISE is similar to that of instances in an instance-based learner. Considering instances to be maximally specific rules unifies instances and rules syntactically; applying rules in the best-match way described unifies instances and rules semantically, completing the unification of the two.

3.3 Search Procedure and Evaluation

Unlike conventional rule induction algorithms, RISE does not construct one rule at a time, but instead induces all rules in parallel. In addition, heuristic evaluation is not performed for each rule separately, but for the whole rule set at once. Changes to an individual rule are evaluated in terms of their effect on the global accuracy of the rule set. This "conquering without separating" strategy differs markedly from the earlier "separate and conquer" one; the aim is to attenuate the fragmentation problem as much as possible. Another major difference is that RISE's direction of search is specific-to-general. Rules are generalized by dropping conditions on symbolic attributes, and broadening intervals for numeric ones. This is not done one attribute at a time, but rather by a clustering-like approach: each rule repeatedly finds the nearest example of its class that it does not yet cover, and attempts to minimally generalize itself to cover it. If the effect of this on global accuracy is positive, the change is retained. This process stops when no further change causes any improvement. The initial rule set is the training set itself (i.e., each instance is a candidate rule). In the worst case no generalizations are accepted, and the final rule set is still the training set, leading to a pure instance-based algorithm. At the other extreme, the rules generated may completely cover the instance space (or all regions of it with non-zero probability),

Table 3.1. The RISE algorithm.

Input: ES is the training set.

```
Procedure RISE (ES)
Let RS be ES.
Compute Acc(RS).
Repeat
   For each rule R in RS,
      Find the nearest example E to R not already covered by it,
         and of R's class.
      Let R' = Most_Specific_Generalization(R, E).
      Let RS' = RS with R replaced by R'.
      If Acc(RS') ≥ Acc(RS)
         Then Replace RS by RS',
               If R' is identical to another rule in RS,
                  Then Delete R' from RS.
Until no increase in Acc(RS) is obtained.
Return RS.
```

leading to a pure rule induction algorithm. Thus RISE has pure IBL and pure rule induction as special cases of its behavior, either of which will be produced if the search procedure described finds it appropriate. More generally, however, the final rule set may contain some ungeneralized exemplars as well as more abstract rules, leading to a broad spectrum of behavior between the two extremes.

Table 3.1 summarizes this process in pseudo-code. In the course of generalization two rules may become identical, in which case they are merged. In each cycle the new rule set is adopted even if its apparent accuracy is the same as the old one's. This is a direct application of Occam's razor: when two theories appear to perform identically, prefer the simpler one.

Table 3.2 shows in pseudo-code how a rule is minimally generalized to cover an example previously outside its scope. In a nutshell, all conditions on symbolic attributes that are not satisfied by the example are dropped, and all intervals are extended to include the example attribute's value at the border, if necessary. This is indeed the most specific generalization that will work, given the representation language used (e.g., internal disjunction is not permitted). Missing values are treated in a fashion consistent with the definition of distance above: a missing numeric value is considered to match any other value, and a missing symbolic value in the example or in the rule (but not both) causes the corresponding condition to be dropped.

The *accuracy* $Acc(RS, ES)$ of a rule set RS on a set of examples ES is defined as the fraction of those examples that it correctly classifies. A rule set

Table 3.2. Generalization of a rule to cover an example.

Inputs: $R = (A_1, A_2, \ldots, A_a, C_R)$ is a rule, $E = (e_1, e_2, \ldots, e_a, C_E)$ is an example. A_i is either True, $e_i = r_i$, or $r_{i,lower} \leq e_i \leq r_{i,upper}$.

Function Most_Specific_Generalization (R, E)
For each attribute i,
  If A_i = True
    Then Do nothing.
    Else if i is symbolic and $e_i \neq r_i$
      Then A_i = True.
    Else if i is numeric and $e_i > r_{i,upper}$
      Then $r_{i,upper} = e_i$.
    Else if i is numeric and $e_i < r_{i,lower}$
      Then $r_{i,lower} = e_i$.

classifies an example correctly when the nearest rule to the example has the same class as it. Whenever the example set ES is simply the whole training set this will be left implicit (i.e., the accuracy will be denoted by $Acc(RS)$). There is no need to use the Laplace correction when comparing the accuracy of different rule sets on a training set, because the denominator of the accuracy (i.e., the number of examples matched) is exactly the same for all rule sets (being the size of the training set).

Accuracy is measured using a leave-one-out methodology: when attempting to classify an example, the corresponding rule is left out, unless it has already been expanded to cover other examples as well. This method would not be efficient if the accuracy of the entire rule set had to be computed from scratch every time an individual change is considered. This would involve repeatedly matching all rules (or all but one) against all examples, leading to a clearly unacceptable time cost. Fortunately, at each step only the change in accuracy $\Delta Acc(RS)$ needs to be considered. Each example memorizes the distance to the nearest rule (i.e., the rule that wins it) and that rule's identification. The memory cost of this is $O(1)$ per example, and is therefore negligible. With this information, all that is necessary when a rule is generalized is to match that single rule against all examples, and check if it wins any that it did not before. Its effect on these is then ascertained. If a previously misclassified example is now correctly classified, the numerator of $Acc(RS)$ is incremented; if the reverse takes place, it is decremented. Otherwise there is no change. If the sum of increments and decrements is greater than or equal to 0, the new rule is adopted, and the relevant structures are updated; otherwise it is rejected.

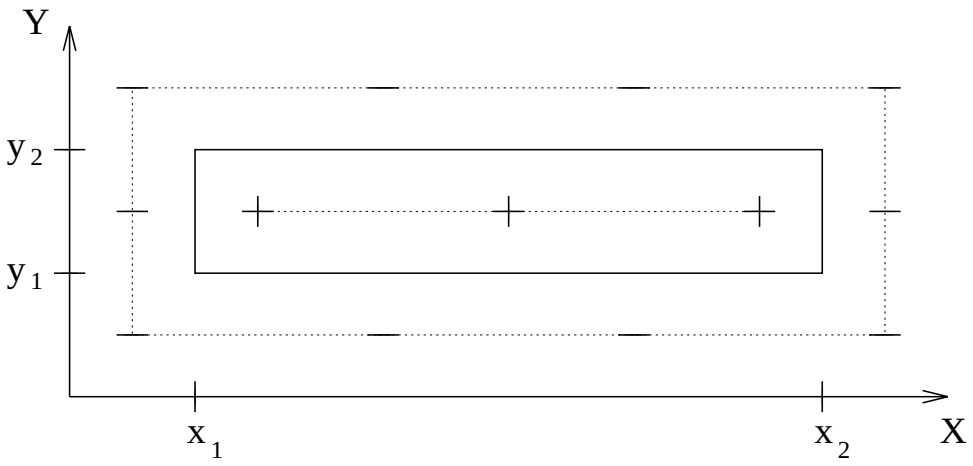

Figure 3.1. Rules formed by RISE for the rectangle concept.

3.4 Three Examples

We can now revisit the two examples described in the previous chapter, and see what RISE will do in each case. A third example will illustrate how RISE combines rule induction and IBL in its inductive behavior.

RISE's solution to the problem of Figure 2.3 is shown in Figure 3.1. RISE generalizes the three positive examples to the straight-line segment shown as a dotted line inside the rectangle: each positive example is first generalized to a segment joining it and the nearest example that is also positive (the middle one, for the left and right examples, and either the left or the right one, for the middle example), and then to a segment joining all three, at which point the three rules coalesce into one. Similarly, the negative examples are generalized to the four segments shown outside the rectangle.

Test examples are now classified according to which segment is nearest, resulting in an almost perfect reproduction of the target rectangle. The only difference is a slight rounding of the corners, reflecting the fact that the set of points at the same distance from a point and a line is a parabola (in this case, the point being the tip of the inner straight-line segment, and the line being either of the outside segments nearest the corner). Notice that, for classification purposes, each segment outside the rectangle is effectively equivalent to the infinite straight line that includes it. Thus attribute X is effectively dropped from the negative instances below y_1 (lower horizontal segment) and above y_2 (upper), and similarly Y is effectively dropped from the negative instances on the left and right. This simple example shows how RISE can (at least in some cases) overcome the context-dependency problems exhibited by instance-based learners.

RISE's solution to the problem of Figure 2.2 is shown in Figure 3.2. The four negative instances in the center are generalized to the square with dotted boundary shown, and similar generalizations are formed for the remaining positive and

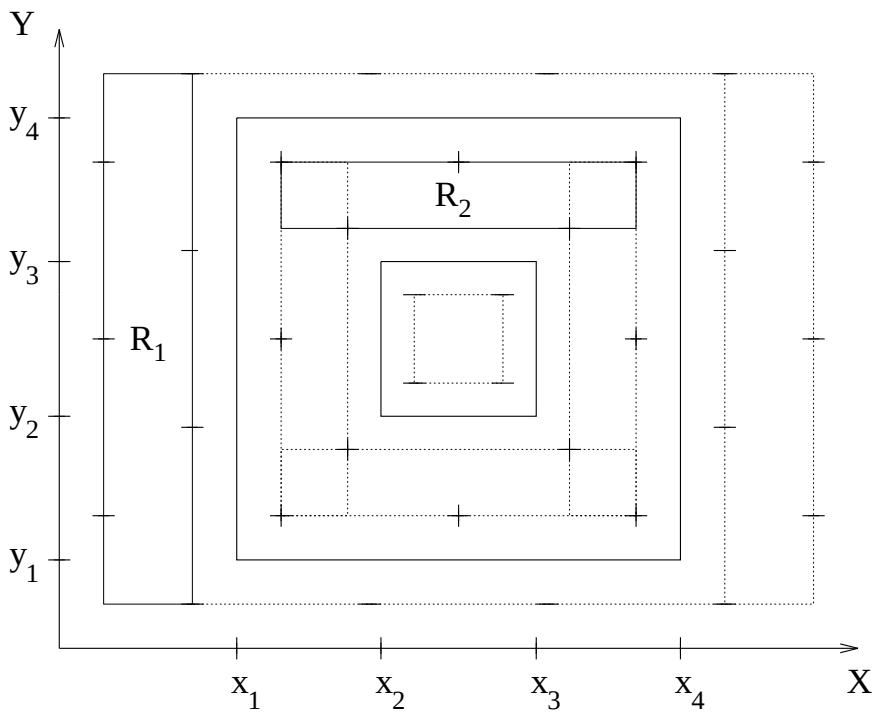

Figure 3.2. Rules formed by RISE for the square doughnut concept.

negative instances. For illustrative purposes, two of the rectangles produced (R_1 and R_2) are shown with solid boundaries. Again, an almost perfect reproduction of the target square doughnut is obtained, with only a slight rounding at the corners (for the same reasons as before). The fragmentation problem is reduced, and no false small disjuncts appear, because each example can "see" all other examples when trying to generalize itself into a rule.

In subsequent chapters, the question of whether or not RISE performs better than IBL and rule induction algorithms for the reasons hypothesized will be examined more systematically, using more realistic target concepts. However, these two examples show in a simple setting how RISE can improve on the behavior of those algorithms.

The following example illustrates RISE's ability to behave as either IBL or rule induction, and to transition smoothly between the two, as well as its ability to induce nonlinear frontiers. Let the instance space be the plane, as before, and consider a training set composed of the three positive and two negative training examples in Figure 3.3. From this training set, a typical rule learner will induce a frontier similar to the one shown in Figure 3.4: a horizontal straight line, with the region below it labeled positive, and the region above it labeled negative. The one-nearest-neighbor classifier with Euclidean distance will produce the zigzag frontier shown in Figure 3.5. RISE will generalize the two negative examples to a straight-line segment with the two examples as the endpoints, and will generalize the three positive examples to another straight line segment, with the rightmost and leftmost

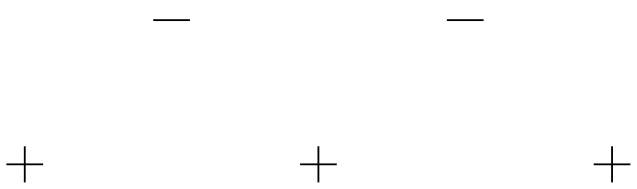

Figure 3.3. A training set.

examples as endpoints. The resulting frontier is shown in Figure 3.6. Between points b and c it is a straight line, identical to the frontier produced by the rule learner. To the left of a it is a diagonal line, identical to the frontier produced by the nearest-neighbor classifier; and similarly to the right of d. Between a and b it is an arc of a parabola with it focus at the leftmost negative example. This arc transitions smoothly between the nearest-neighbor frontier to its left and the rule-induction frontier to its right. Between c and d a similar transition occurs. Thus RISE behaves like an instance-based classifier in some parts of the instance space, and like a rule learner in others; and it transitions smoothly between the two, creating non-linear frontiers in the process.

3.5 Time Complexity of RISE

It is possible to derive an upper bound for the time complexity of RISE, showing that its worst-case efficiency is comparable to that of other rule induction algorithms. Let e represent the number of examples in the training set, a the number of attributes used to describe each, v_s (v_n) the maximum number of observed values per symbolic (numeric) attribute, r the number of rules, and c the number of classes into which the examples fall. Assume for now that all attributes are symbolic. The initialization phase of the algorithm consists of three operations. The first is copying the examples to the rules, and takes $O(ea)$ time. The second

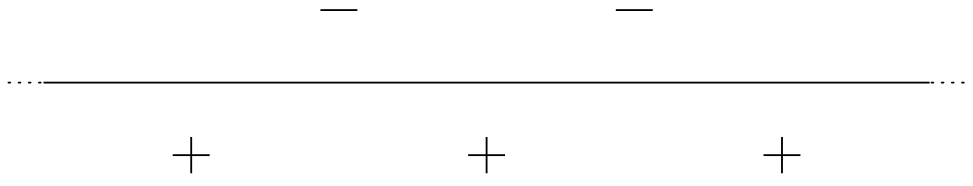

Figure 3.4. Frontier induced by a rule learner.

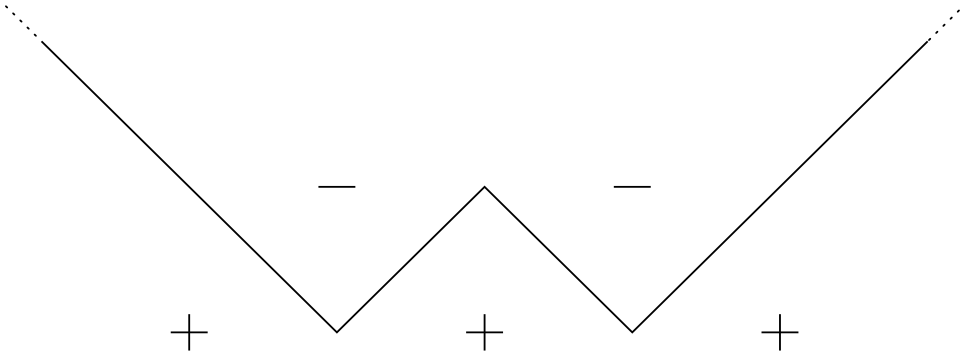

Figure 3.5. Frontier induced by an instance-based learner.

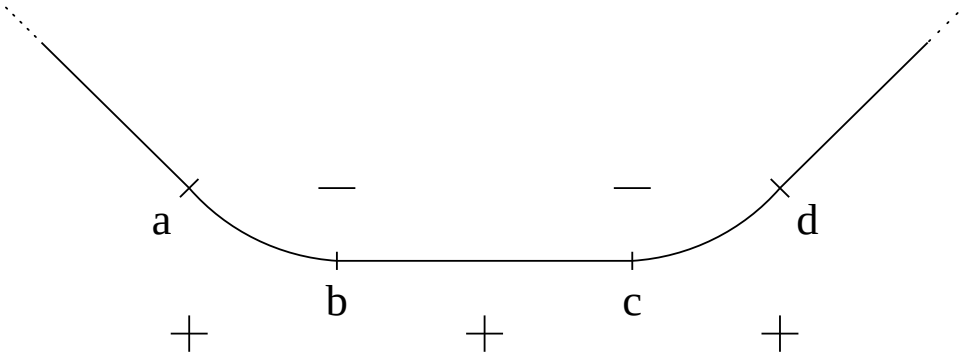

Figure 3.6. Frontier induced by RISE.

is compiling a table of SVDM distances, taking $O(ea + av_s^2c)$ (ea to run through all the examples, for each one noting the correspondence between each attribute's value and the class, and av_s^2c to sum the results for all classes, for each pair of values of each attribute). The third operation is finding each example's closest rule and computing the initial accuracy of the rule set, which involves matching all rules against all examples, and so takes $O(e^2a)$ time. The total time necessary for initialization is therefore $O(e^2a + av_s^2c)$.

The heart of the algorithm consists of four steps: finding a rule's nearest example, generalizing the rule to cover it, comparing the altered rule to all examples to see if any are newly won, and (if the change is adopted) comparing the rule to all other rules to check for duplications. These operations consume respectively $O(ea)$, $O(a)$, $O(ea)$ and $O(ra)$ time, for a total of $O(ea+ra)$. Since each "Repeat" cycle (see Table 3.1) consists of doing this for all r rules, each such cycle takes at worst $O[r(ea + ra)]$ time. In RISE each example produces at most one rule; therefore $r \leq e$, and this time is at worst $O(e^2a)$.

How many "Repeat" cycles can the algorithm perform in the worst case? Two answers are possible, depending on how the stopping criterion is interpreted. If it is applied individually (i.e., generalization of a given rule stops as soon as covering the nearest example produces no improvement), then the "Repeat" cycle is performed at worst $O(a)$ times, since each cycle must remove at least one condition, and a rule contains at most a conditions, this being true for each rule. On the other hand, if the stopping criterion is applied globally (i.e., generalization of a given rule stops only when no change to *any* rule produces an improvement), the "Repeat" cycle can in theory be performed up to $O(ea)$ times, because in the worst case only one condition of one rule will be dropped in each entire cycle, each time causing some currently-unprofitable change in another rule to become profitable in the next round. However, this is extremely unlikely. The two policies were empirically compared (see next chapter), showing no appreciable difference between the two in accuracy or time. Multiplying the values above by the cost of a single "Repeat" cycle yields a total time complexity of $O(e^2a^2)$ or $O(e^3a^2)$ respectively. Since $e \geq v_s$, and assuming that $a \geq c$, which is generally the case, the smaller of these values dominates the complexity of the initialization phase, and both therefore constitute upper bounds on the time complexity of the whole algorithm in their respective situations.

The time complexity of CN2 and AQ-style algorithms is $O(be^2a^2)$, where b is the beam size, an integer-valued internal parameter of the algorithm (Clark & Niblett, 1989).[1] The worst-case complexity of growing a decision tree is $O(ea^2)$ when only symbolic attributes are present (Utgoff, 1989b), but becomes quadratic or worse in e with numeric ones, mainly due to the need for repeated sorting operations (Catlett, 1991). The final pruning stage used in C4.5 and several rule

[1] The computations in this reference are only for the basic step of the algorithms, which is embedded in loops that may run $O(ea)$ in the worst case, leading to the bound shown.

induction systems may also in general be worse than quadratic in e (Cohen, 1995). Thus RISE's worst-case time complexity is comparable to that of other rule and decision tree learners.[2] Average-case time is also likely to be substantially smaller than the worst case, because some of the assumptions above are overly pessimistic (e.g., in general r will seldom remain equal to e, but will instead shrink rapidly; attributes will be dropped several at a time, and not all will be removed; increasing e may not increase the number of "Repeat" cycles, even with a global stopping criterion; etc.).

The introduction of numeric values simply increases the values above by a factor of v_n, since the single-step removal of a condition may now be replaced by at most $O(v_n)$ steps of expanding the corresponding interval. Again, in practice only a small number of steps may actually be required. However, in the case of real-valued attributes for which arbitrarily fine distinctions are possible, and therefore as many different values as there are training examples may occur, this can lead to some inefficiency. This problem is only potentially significant in large datasets, since the number of observed values of an attribute is bounded from above by the number of training examples. To obviate it, in datasets with more than 3000 examples the generalization of a rule to cover an example (Table 3.2) is adapted as follows. When extending an interval to include an example value above it (or below), the interval's upper (lower) limit is not set to the example's value, but to that value plus (minus) a delta which is set by default to 5% of the maximum range the value was observed to vary over. In addition to potentially reducing the running time of the algorithm, this policy can have a positive effect on its accuracy, by avoiding overfitting. This was indeed observed in the empirical study described below.

RISE 3.1 has been optimized with respect to running time in two additional ways: pruning and windowing. Pruning is performed by sometimes cutting short the computation of the distance between a rule and an example; this is akin to reducing search by means of branch-and-bound techniques. This is possible in two situations (see Table 3.1). When a rule searches for its nearest example, it can discard a candidate as soon as its distance becomes larger than the shortest one found so far. Similarly, when a tentatively-generalized rule searches for the examples it now wins, its distance to each example needs to be computed only until it becomes larger than the current winning rule's one. This form of pruning has no effect on the algorithm's output, and in general will also not change its worst-case time complexity, but can significantly reduce its average running time.

In datasets with more than 3000 examples, windowing is used. It is applied to RISE in a fashion similar to C4.5's (Quinlan, 1993a), and proceeds as follows. Initially, only $2\sqrt{e}$ examples randomly extracted from the training set are used for learning. This sample is stratified (i.e., it contains approximately equal proportions

[2] Recently, a number of algorithms have been proposed with the specific goal of reducing this time complexity while maintaining accuracy (Fürnkranz & Widmer, 1994; Cohen, 1995). Chapter 7 describes one such algorithm based on RISE.

of all classes); this makes it possible to still learn classes that have few representatives in the original training set. If the remaining training examples are correctly classified by the resulting rule set, this set is output. Otherwise, the misclassified examples are added to the initial example set, and this process repeats until it produces no improvement in accuracy on two successive expansions. This policy of requiring two successive failures to stop has been verified empirically to lead to better results in the case of RISE than the policy followed by C4.5, of stopping as soon as there is no improvement in accuracy. The latter is more prone to premature stopping (i.e., stopping at a local minimum of the accuracy improvement curve).

In the best case, only $O(\sqrt{e})$ examples are used, and the algorithm becomes linear in the training set size. In the worst case, the window grows to include the entire training set (or nearly so), and the process is more costly than learning directly on that set. This is particularly likely in noisy domains, where it has been observed to lead to serious performance degradation in the case of C4.5 (Catlett, 1991). To avoid this, the implementation used in RISE also limits the number of times the window is grown to a prespecified maximum (5 by default). This should help prevent the system from attempting to fit the noise in domains where this is a problem, and has been found empirically to sometimes achieve large reductions in running time compared to the unlimited-expansion version, without seriously affecting accuracy. The use of windowing and other methods to speed up RISE is further discussed in Chapter 7.

3.6 Large-Sample Properties of RISE

Although it is the small-sample behavior of learning algorithms that is of most practical interest, it is still comforting to know that a learner behaves "well" in the large-sample limit. To make this notion of "good behavior" more precise, the following definitions are needed. The *large-sample error rate* of a classifier is the limit its error rate approaches when the sample size tends to infinity, if such a limit exists. The *Bayes rate* ϵ^* for a domain is the lowest error rate achievable by any classifier in that domain (Duda & Hart, 1973). The *Bayes classifier* for a domain is the classifier that achieves the Bayes rate in that domain. Thus, no classifier can be more accurate than the Bayes classifier. However, given a large enough sample, it should be possible to get close to this limit. The sample size required for this may be very large; even millions or billions of examples is not necessarily enough. An algorithm that is "well behaved" in the large-sample limit is one whose large-sample error rate is guaranteed (with probability one) to be close to the Bayes rate, for example by being only a small multiple of it. Large-sample convergence properties have been proved for nearest-neighbor algorithms (Cover & Hart, 1967) and decision-tree learners (Gordon & Olshen, 1984), but so far not for "separate and conquer" rule learners. However, because of RISE's

relation to IBL, properties similar to those of nearest-neighbor algorithms can be derived for it, as follows.

Let ϵ_{NN} be the large-sample error rate of the nearest-neighbor classifier, and c be the number of classes. Then $\epsilon^* \leq \epsilon_{NN} \leq \epsilon^*(2 - c\epsilon^*/(c-1))$ (Cover & Hart, 1967). In other words, the large-sample error rate of the nearest-neighbor classifier is at most twice the Bayes rate. This result applies in metric spaces, which includes numeric domains when Equation 2.4 is used as the metric, and also includes symbolic or mixed domains when the overlap metric (Equation 2.5) is used. Let RISE$_O$ be RISE using the overlap metric for symbolic attributes, and Equation 3.3 for numeric ones. Then the result above also holds for RISE$_O$. More precisely, if ϵ_{RISE_O} is the large-sample error rate of RISE$_O$, and c is the number of classes, then $\epsilon^* \leq \epsilon_{RISE_O} \leq \epsilon^*(2 - c\epsilon^*/(c-1))$. This is due to the following (see Table 3.1). Initially (before the "Repeat" cycle begins) RISE$_O$ acts as a pure nearest-neighbor classifier, so its error rate is the same as nearest-neighbor's. Thereafter, each generalization step is performed iff it does not decrease RISE$_O$'s accuracy measured on the training sample. However, with a large enough sample, this estimate becomes arbitrarily close to RISE$_O$'s true accuracy, by the central limit theorem. Thus each generalization step is performed iff it does not decrease RISE$_O$'s true accuracy. Since RISE$_O$ is initially as accurate as nearest neighbor and each generalization step can only maintain or increase its accuracy, RISE$_O$'s final accuracy is greater than or equal to that of nearest neighbor, by induction. Thus the upper bound for nearest-neighbor applies also to RISE$_O$. The lower bound applies by definition of Bayes rate.

It is not possible to prove a similar result when using the SVDM measure (Equation 2.6), since in this case the metric assumption that $\forall E_1, E_2 \ (E_1 \neq E_2 \Rightarrow \Delta(E_1, E_2) > 0)$ does not hold, and RISE's or nearest neighbor's error will be that of random guessing if no individual attribute values are correlated with the class. For example, when attempting to learn the Boolean parity function (1 when an even number of attributes is 1, and 0 otherwise), in the large-sample limit the SVDM distance between the 0 and 1 values of any attribute is zero, since either class occurs 50% of the time given either value. Thus the distance between all examples is zero, and the classifier's performance is equivalent to random guessing.

RISE$_O$'s large-sample convergence can also be understood by decomposing predictive error into bias and variance (Kong & Dietterich, 1995; Kohavi & Wolpert, 1996; Tibshirani, 1996; Breiman, 1996b; Friedman, 1996). Since RISE$_O$ includes nearest neighbor as a special case, RISE$_O$'s bias cannot be greater than the latter's. Thus RISE$_O$'s error can only be greater than nearest neighbor's if its variance is greater. However, in the large-sample limit the variance becomes zero, and so in this limit RISE$_O$ cannot be less accurate than nearest neighbor.

3.7 Related Work

Recent years have seen much research combining multiple learning paradigms. The crucial conceptual difference between RISE and this work is that RISE *unifies* rather than *combines* its parent approaches. Typical multistrategy learning systems include the component learners as subprocedures, resulting in a learner that is more complex than the sum of the individual algorithms. RISE, in contrast, is a single algorithm, that is as simple as its parents (or simpler). This has both scientific and technical advantages. The model of learning embodied in RISE is more parsimonious than that of a combined learner, and leads to a deeper understanding of the relations between the unified paradigms. Because of its simplicity, the algorithm is easier to understand and implement than a combined learner and its subprocedures.

Perhaps the earliest rule induction system to employ best-match classification was AQ15 (Michalski, Mozetic, Hong & Lavrac, 1986); a more recent version is AQ17-HCI (Wnek & Michalski, 1994). It differed from RISE in that it was a general-to-specific, separate-and-conquer system that learned purely logical rules, and only introduced the best-match policy in a post-processing step, with the goal of removing rules from the set without adversely affecting accuracy. It also used a different distance measure. Its approach is carried further in the FCLS system (Zhang, 1990), which combines rules with exemplars in an attempt to alleviate the small disjuncts problem. Unlike RISE, FCLS employs different representations for rules and exemplars, and it uses the separate-and-conquer strategy of its AQ ancestors.

RISE addresses the fragmentation problem in rule induction by employing a "conquering without separating" induction strategy. Other approaches to this problem include constructing new attributes (Pagallo & Haussler, 1990) and converting decision trees to decision graphs (Oliveira & Sangiovanni-Vincentelli, 1995; Kohavi & Li, 1995).

Viewed as an instance-based learner, RISE performs context-sensitive feature selection, which can be seen as an extreme form of context-sensitive feature weighting. A number of methods of this type have been proposed in the literature (Aha & Goldstone, 1992; Hastie & Tibshirani, 1996; Atkeson, Moore & Schaal, 1997). These are reviewed in more detail in Chapter 6. Non-metric, context-sensitive distance measures for IBL are discussed in (Biberman, 1994). Another important aspect of RISE is its policy for combining numeric and symbolic attributes, using SVDM for the former and Euclidean distance for the latter. Alternative approaches have been explored by Ting (1994) and Mohri and Tanaka (1994).

MCS (Brodley, 1995) has perhaps the most similar aims to RISE's, but uses an entirely different approach (applying meta-knowledge to detect when one algorithm should be used instead of another), and combines instead decision trees

with IBL and linear discriminant functions. Golding and Rosenbloom (1991) designed a system that gainfully combined case-based and rule-based reasoning, but it did not learn, it matched rules exactly, and it used different representations and match procedures for cases and rules, instead of RISE's unified approach. Quinlan (1993b) has successfully combined IBL with trees and other methods, but for the purpose of regression as opposed to classification, performing this combination only at classification time, and in a way that depends critically on the predicted value being continuous. Several induction algorithms proposed in the literature can be seen as empirical multi-strategy learners, but combining different paradigms from RISE's: decision trees and rules (Quinlan, 1987a), decision trees and perceptrons (Utgoff, 1989a), rules and Bayesian classification (Goodman, Higgins, Miller & Smyth, 1992), backpropagation and genetic algorithms (Belew, McInerney & Schraudolph, 1992), instances and prototypes (Scott & Sage, 1992), decision trees and kernel density estimation (Smyth, Gray & Fayyad, 1995), decision trees and simple Bayesian classifiers (Kohavi, 1996), etc. (see (Michalski & Wnek, 1996) for several recent examples).

In form, the most similar system to RISE in the literature is EACH (Salzberg, 1991), which generalizes instances to hyperrectangles, and classifies each test example according to its nearest hyperrectangle. Its measure of the distance between an example and a hyperrectangle is similar to Equation 3.3. EACH differs from RISE in many ways: it is applicable only in purely numerical domains, is an incremental algorithm, never drops attributes, uses different heuristics and search strategies, always prefers the most specific hyperrectangle, etc. Recently Wettschereck and Dietterich (1995) carried out a detailed comparison of EACH and k-nearest-neighbor (kNN), and designed an algorithm that combines the two (Wettschereck, 1994), but does not achieve greater accuracy than kNN alone. They found EACH to be less accurate than kNN in most of the domains studied, and the chief cause of this to be EACH's use of overlapping rectangles. Since RISE and EACH use similar representations in the case of numeric attributes, this warrants closer examination.

Two issues are involved in Wettschereck and Dietterich's study, and they should be clearly distinguished. One is whether using a single nearest neighbor is preferable to using several. This has been studied for many years in the nearest neighbor literature (e.g., (Cover & Hart, 1967)). Another issue, of more interest here, is whether or not generalization of instances to hyperrectangles is beneficial. Unfortunately, the study does not decouple the two issues, and it is not possible to determine which of the two factors (or both) is responsible for the observed differences in performance. EACH (and RISE) can be extended to use the k nearest rules for classification; doing so and comparing the resulting approaches with kNN is a worthy topic for future research. Here we will discuss our results on the issue of whether or not to generalize instances to rules, in the context of using the single nearest rule for classification.

In the cross-validation studies reported in the next chapter, RISE's tie-breaking policy based on Laplace accuracy was compared with one selecting the

most specific rule as in EACH, and found to be clearly superior. This can be understood as follows. In regions of overlap, EACH arbitrarily assigns all examples to the class of the most specific hyperrectangle. In contrast, RISE's learning strategy approximates the optimal decision rule of placing the boundary between two classes at the point where the density of examples from one overtakes that of the other (Duda & Hart, 1973). This is because a rule is started from each example, and its generalization halts when it would include more examples of other classes than of the example's one. When rules overlap, the use of Laplace accuracy implies that, given similar-sized samples, each rule prevails in regions where the density of examples of its class is greater. RISE's batch-learning approach also avoids the problems that EACH's incremental learning one was observed to suffer from. As reported in the next chapter, RISE outperformed two one-nearest-neighbor algorithms (PEBLS and RISE's own IBL component) in a large-scale empirical study. All these facts support the conclusion that generalizing instances to rules can indeed produce substantial improvements in accuracy, if done in an appropriate manner.

3.8 Summary

The RISE algorithm differs from most multistrategy learners in that it unifies, rather than combines, its parent approaches. RISE employs a uniform representation for rules and instances, by treating instances as maximally specific rules and applying rules in a best-match fashion. RISE employs a specific-to-general, "conquering without separating" search strategy, in which rules are learned by gradually generalizing instances until no improvement in accuracy is obtained. This allows it to combat the fragmentation and small disjuncts problems that previous rule induction approaches suffer from. RISE is also able to selectively generalize and drop attributes from instances, and thus to combat the context-dependency problems that can affect instance-based learners. Theoretical analysis shows that RISE, properly optimized, is as efficient in the worst case as other induction algorithms, and that RISE using the overlap metric has a large-sample error rate of at most twice the Bayes rate.

Chapter 4
Empirical Evaluation of RISE

4.1 Overview

An extensive empirical study was carried out with the goals of refining RISE, comparing its performance to that of previous approaches, and determining the role of its main components in that performance. This chapter describes the characteristics and reports the results of this study.

4.2 Application Databases

Thirty databases from the UCI repository (Merz, Murphy & Aha, 1997) were used in the study. An attempt was made to include every available dataset that has been widely used in inductive learning studies, and beyond that to provide a wide sampling of: symbolic, numeric and mixed datasets; small, medium and large datasets; datasets of varying difficulty, as expressed in the highest accuracy previously achieved; and datasets from a wide range of application areas. Reasons for excluding datasets in the UCI repository from the study were:

- Some datasets are inappropriate for this type of algorithm. This includes: datasets involving relational data, domain theories, structured instances, or variant instances; datasets intended for psychological studies, and therefore minuscule; and datasets intended for regression tasks (i.e., where the predicted value is numeric).
- In some cases there are multiple datasets from the same domain, and duplicate datasets. In this case only the most widely used dataset was included.
- There are many more purely numerical datasets than symbolic or mixed datasets in the repository. Use of all would result in a higher proportion of this type of dataset than is usually the case in machine learning studies, undermining comparisons with previous literature.
- Some datasets are inadequately documented and/or formatted.
- Some datasets are artificially generated ones, without correspondence to any real-world problem.

- In some datasets there is no clear feature to use as the class.

Essentially all the datasets in the UCI repository at the time of this study that did not fall under one of these restrictions were included. Table 4.1 lists these datasets in alphabetical order of code-name, and summarizes some of their main characteristics. Datasets included in the listing of empirical results in (Holte, 1993) are referred to by the same codes. The filenames of the datasets used, and any conversions that had to be performed, are listed in Appendix A.

4.3 Variations on RISE

Once the main features of an algorithm have been laid down, many detailed design decisions still need to be made. Accordingly, in the first phase of the empirical study, the first 15 datasets in Table 4.1 (from breast cancer to wine) were used to compare different versions of RISE, and select the best one by 10-fold cross-validation. In this procedure, the dataset is randomly divided into 10 equal-sized subsets, and 10 runs of each version of the algorithm are carried out. In each of the 10 runs a different subset of the data is left out, and training is carried out on the remaining nine. The accuracy of each version of the algorithm is then measured on the subset that was left out, and the average accuracy for the 10 runs is computed.

For the sake of conciseness, tables containing numerical results for each comparison are omitted, but the main observations are of interest, not only to RISE but also in the wider context of the issues they address, and are therefore summarized below. In each case, only the general trend is reported; almost invariably, exceptions to it were also observed. Whenever no significant difference in accuracy was observed, the simpler version of the algorithm was chosen; otherwise the more accurate one prevailed. In general, differences in accuracy (averaged over all datasets) of less than 1% were not considered significant. The comparisons made and respective conclusions were as follows.

- *Use of weights on exemplars.* Two versions were compared: no weights, and each rule weighted by the inverse of its Laplace accuracy, leading unreliable rules to appear farther from new instances. This is similar to the weighting scheme used in PEBLS (Cost & Salzberg, 1993), but slightly more sophisticated because the Laplace correction is used. Note that this correction is indeed justified when dealing with individual rules. No significant difference was observed. This may be due to the fact that the datasets are not too noisy, to the fact that RISE's basic approach to induction is successful by itself in combatting noise, or to both. No weights is the default in RISE 3.1.
- *Tie-breaking.* When more than one rule was equally near the test example, ties were broken by choosing the one with highest Laplace accuracy, by choosing the most specific one, and by frequency-based voting as done in

Table 4.1: Datasets used in the empirical study. The columns are, in order: name of the dataset; 2-letter code used to refer to it in subsequent tables; number of examples; number of attributes; number of numeric attributes; number of classes; percentage of missing values; and whether or not the dataset includes inconsistent examples (i.e., identical examples with different classes).

Dataset	Code	Exs.	Atts.	Num.	Classes	Missing	Incons.
Breast cancer	BC	286	9	4	2	0.3	Yes
Credit screening	CE	690	15	6	2	0.6	No
Chess endgames	CH	3196	36	0	2	0.0	No
Pima diabetes	DI	768	8	8	2	0.0	No
Hepatitis	HE	155	19	6	2	5.7	No
Iris	IR	150	4	4	3	0.0	No
Labor negotiat.	LA	57	16	8	2	35.7	No
Lung cancer	LC	32	56	0	3	0.3	No
Liver disease	LD	345	6	6	2	0.0	No
Contact lenses	LE	24	4	0	3	0.0	No
Lymphography	LY	148	18	3	4	0.0	No
Primary tumor	PT	339	17	0	21	3.9	Yes
Soybean	SO	47	35	0	4	0.0	No
Voting records	VO	435	16	0	2	5.6	No
Wine	WI	178	13	13	3	0.0	No
Audiology	AD	200	69	0	24	2.1	No
Annealing	AN	798	38	9	5	64.9	No
Echocardiogram	EC	131	7	6	2	4.4	Yes
Glass	GL	214	9	9	6	0.0	No
Heart disease	HD	303	13	6	2	0.2	No
Horse colic	HO	300	22	7	2	24.3	Yes
Thyroid disease	HY	3163	25	7	2	6.7	Yes
LED	LI	100	7	0	10	0.0	Yes
Mushroom	MU	8124	22	0	2	1.4	No
Post-operative	PO	90	8	8	3	0.4	Yes
DNA promoters	PR	106	57	0	2	0.0	No
Solar flare	SF	323	12	3	6	0.0	Yes
Sonar	SN	208	60	60	2	0.0	No
Splice junctions	SP	3190	60	0	3	0.0	Yes
Zoology	ZO	101	16	1	7	0.0	No

CN2. The best-performing alternative was Laplace accuracy. Use of specificity had a clear negative effect on accuracy, contradicting heuristics used in some other systems (e.g., EACH (Salzberg, 1991)). Tie-breaking when the accuracies are also identical was described in the previous chapter.

- *Distance computation.* Values of $q = 1$ and $q = 2$ (see Equation 2.6) were tried, each combined with values of $s = 1$ and $s = 2$ (see Equation 3.1). No appreciable difference was observed, confirming previous results (Cost & Salzberg, 1993). The default values for RISE 3.1 are $q = 1$ and $s = 2$ (Euclidean distance).
- *Treatment of numeric values.* The following versions were compared: normalization by the attribute's observed range, as in Equation 3.3; normalization by 3 and 4 times the standard deviation for the attribute; discretization into equal-sized intervals up to a maximum of 10; and intervalization by entropy minimization, i.e., ordering the values and successively choosing the splitting point that most reduces the entropy, until a maximum number of intervals is reached (10, 100) or the reduction obtained is at or below a given minimum (10%, 1%, 0%). The latter approach is similar to Catlett's (1991).[1] The two first methods were in general clearly superior to the latter two, although this was reversed in some datasets. The entropy-based method also caused a noticeable increase in computation time for the larger datasets. The two types of normalization performed very similarly; range was chosen, due to its greater simplicity.
- *Treatment of missing values.* Missing symbolic values were treated in two ways: as legitimate symbolic values (see earlier discussion), and matching every value as done for numeric values. The first alternative proved superior.
- *Search.* Two types of search were attempted: finding only the nearest example and attempting to cover it, and finding the 3 nearest examples, attempting to cover each and choosing the best result (or no change, as before). The latter alternative produced no substantial improvement, as well as being predictably slower. The first was chosen.
- *Final search.* Two alternatives were tried when the generalization of a rule to the nearest example does not improve accuracy. One was to do nothing. The other was to attempt generalization to all other examples of the rule's class in order of increasing distance from it, until an improvement was obtained or the examples were exhausted. Again, this variation produced a slowdown and no overall improvement in accuracy, and was not adopted.
- *Stopping criterion.* Two stopping criteria were compared: local, where a rule's generalization is terminated as soon as an attempt to generalize it fails, and global, where this termination only occurs when such attempts have failed for all rules. There was no significant difference between the

[1] A related approach using the minimum description length principle is described in (Fayyad & Irani, 1993).

two in running time, and global stopping tended to produce slightly higher accuracies, so this policy was chosen.

- *Merging rules.* Three policies for merging rules were compared: deleting duplicate rules, deleting subsumed rules (i.e., rules logically implying the subsuming rule), and deleting subsumed rules only if they were not more accurate than the subsuming rule. The rationale for the last policy is that in RISE a subsumed rule can have a positive effect on overall accuracy, because it may win examples that the subsuming rule would not. The two latter approaches produced some speedup and somewhat more compact rule sets, but also had a noticeable negative effect on accuracy. The default for RISE is therefore deleting only duplicated rules.
- *Simplification.* Two post-processing techniques that further simplified the final rule set were tested, individually and in combination. One was to delete all rules that won no examples. The other was to delete all attributes in a rule that did not have different values in examples of other classes. Both strategies were successful in simplifying the rule sets, particularly the first, with the greatest reductions predictably produced by the combination of the two; compression rates in excess of 90% for the whole algorithm were common. However, this simplification was accompanied by a small overall decrease in accuracy, and post-processing is thus not the default in RISE. These observations stand in contrast to studies on decision tree learners (Quinlan, 1987b) and rule induction systems (Michalski, Mozetic, Hong & Lavrac, 1986), where simplification was accompanied by an increase in accuracy. Together with the results in (Murphy & Pazzani, 1994), (Webb, 1996) and (Domingos, 1997b) that suggest the most accurate decision tree is not always the simplest, this may indicate that the relationship between accuracy and simplicity is not as simple as is sometimes assumed (e.g., (Blumer, Ehrenfeucht, Haussler & Warmuth, 1987; Cheeseman, 1990)).

4.4 Other Systems

In the second part of the empirical study, RISE was compared with a representative of each of its parent approaches: PEBLS for IBL (Cost & Salzberg, 1993), and CN2 for rule induction (Clark & Niblett, 1989). PEBLS is a state-of-the art system, as opposed to the skeleton nearest-neighbor implementations typically used in empirical comparisons. PEBLS 2.1's inability to deal with missing values was overcome by grafting onto it an approach similar to the one selected for RISE (see previous chapter). A recent version of CN2 (6.1) was used, one incorporating Laplace accuracy and unordered rules (Clark & Boswell, 1991). To gauge its position in the overall spectrum of induction methods, RISE was also compared with a system that learns rule sets by way of decision trees, C4.5/C4.5RULES (Quinlan, 1993a). The default classifier (always choosing the most frequent class) was also

included in the study to provide a baseline. Backpropagation (Rumelhart, Hinton & Williams, 1986), although a widely-used learning algorithm, was left out because its need for extensive fine-tuning and very long running times would make a large-scale study of this type difficult to carry out.

To ensure a fair comparison, the 15 datasets used in the first phase of the study to select among different versions of RISE were also used to fine-tune the other algorithms, again choosing the most accurate version of each by 10-fold cross-validation. Since a complete factor analysis would be too expensive, the relevant parameters were instead varied one at a time, starting with the ones thought to be most critical, and selecting the best value for each parameter before going on to the next one. When no clear differences were present, the default was retained. In PEBLS, the number of intervals for each numeric attribute has to be set by the user, and it is recommended that it be kept small for best results. The policy was adopted of using the number of observed values up to a maximum of m, and the optimum m found was 15. All other parameters retained their default values. In CN2, the only non-default choice produced was that of accuracy instead of Laplace accuracy for the evaluation function.[2] In C4.5, a major question is whether to output trees or rules. C4.5RULES produced slightly better results, and was the version chosen. It also has the advantage of being the one most directly comparable to RISE. The non-default choices made were: use windowing (growing 10 trees, the default), require a minimum of 4 examples (instead of 2) in two branches of a test, and use a confidence level of 37.5% for rule pruning (instead of 25%).

The fine-tuned algorithms were then tested on the remaining 15 datasets in Table 4.1 (from audiology to zoology). Note that this procedure is somewhat unfavorable to RISE, since some of these datasets were previously used in the development of the other algorithms, as reported in the references above. Each dataset was randomly divided 50 times into a training set containing two-thirds of the examples, and a testing set containing the remainder. All of the algorithms were trained on each of the 50 training sets and tested on the corresponding testing set. The results reported are averages of these 50 runs. For the sake of completeness, the algorithms were also rerun in these conditions on the 15 tuning datasets.

4.5 Accuracy Comparisons

The average accuracies and sample standard deviations obtained are presented in Table 4.2. The first half of the table shows results on tuning datasets, and the second half shows results on test datasets. Superscripts indicate confidence levels for the difference between RISE and the corresponding algorithm, using a one-tailed paired t test. The t test is appropriate because the accuracies being compared, being means of random samples, are normally distributed, according to the

[2]Clark and Boswell (1991) observed Laplace accuracy to outperform entropy in the context of CN2, but conducted no comparison of Laplace accuracy with uncorrected accuracy.

central limit theorem (DeGroot, 1986), and the variances are unknown and also being estimated; the one-tailed test is preferred over the two-tailed one because the goal is to determine in each case whether RISE is better than the corresponding algorithm, not just whether the two are different. The more sensitive paired test is made possible by, in each run, testing all the algorithms on the same sample. Since 120 individual significance tests are reported, it is possible that some of the differences reported as significant are in fact not so (for example, with 100 tests we can expect 5 non-significant differences to be reported as significant at the 95% confidence level by chance). However, most of the confidence levels obtained are very high, making this effect unlikely. [3]

Although many results in individual datasets are interesting in themselves, these tables are more easily understood if their results are summarized in a few global measures of comparative performance. Five such measures and their values are presented in Table 4.3. They can be described and interpreted as follows.

- *Number of wins.* The first and most obvious test is simply to count the number of datasets where RISE achieved higher average accuracy than the second algorithm, count those where its accuracy was lower, and compare the two (ties, which rarely occur, are not counted either way). The first line of Table 4.3 shows this: for example, RISE performed better than PEBLS in 10 test datasets, and worse in 4 (there was 1 draw). As can be seen, RISE outperformed every other algorithm in this respect by a ratio of roughly 2 to 1.
- *Number of significant wins.* The previous measure is clearly imperfect, for some of the differences can be very small and of no real significance. A good alternative is then to count only those datasets where the difference was significant at a confidence level of 95% or higher (see Table 4.2). This yields roughly similar ratios, with all values somewhat reduced as might be expected, confirming the previous observation that RISE performs substantially better than every other algorithm.
- *Wilcoxon test.* The goal of machine learning is not in general to produce algorithms targeted to one specific domain, but rather to produce algorithms that are accurate across broad classes of domains. In trying to answer the question "Is RISE a more accurate system?" it is then useful to look at the 15 (or 30) datasets used in this study as a sample of size 15 (or 30)

[3] These confidence levels should be interpreted with caution, due to the t test's assumption of independently drawn samples. Thus a 99% level for a dataset means that RISE can be expected with high confidence to outperform the corresponding algorithm on training sets drawn at random from that dataset, since the accuracy results were obtained by independently drawing training sets from the dataset. This is useful for cross-checking the results of this study with previous ones on the same datasets. However, no conclusions can be drawn regarding different datasets drawn at random from the same domain the UCI dataset was, because with respect to the domain the training sets used here are not independent, being overlapping subsets of the same dataset. See (Dietterich, 1996) for more on this issue.

Table 4.2: Empirical results: average accuracies and standard deviations. Superscripts denote confidence levels: 1 is 99.5%, 2 is 99%, 3 is 97.5%, 4 is 95%, 5 is 90%, and 6 is below 90%.

Dataset	RISE	Default	PEBLS	CN2	C4.5
BC	67.7±4.5	68.5±6.8^{6}	64.9±4.0^{1}	65.9±4.7^{1}	67.9±5.1^{6}
CE	83.3±2.4	56.7±3.7^{1}	81.6±2.0^{1}	82.3±2.2^{1}	84.5±2.5^{1}
CH	98.2±0.6	52.3±1.8^{1}	96.8±0.7^{1}	98.7±0.7^{1}	99.2±0.2^{1}
DI	70.4±2.5	65.4±2.4^{1}	70.9±2.6^{6}	73.6±2.4^{1}	73.1±2.5^{1}
HE	78.3±5.7	79.1±3.6^{6}	80.3±5.1^{3}	80.9±3.6^{1}	81.0±5.3^{2}
IR	94.0±2.8	26.3±3.8^{1}	93.2±3.5^{4}	90.8±4.4^{1}	93.5±2.8^{6}
LA	87.2±7.8	66.1±10.2^{1}	91.1±5.1^{1}	82.8±7.6^{1}	81.4±6.1^{1}
LC	44.7±16.4	24.7±15.5^{1}	42.0±15.2^{6}	32.7±13.6^{1}	44.5±14.1^{6}
LD	62.4±4.6	57.8±3.2^{1}	61.2±4.2^{5}	66.8±4.7^{1}	65.1±4.5^{1}
LE	77.2±12.8	60.8±13.1^{1}	72.5±12.6^{1}	67.8±12.4^{1}	63.7±18.4^{1}
LY	78.7±5.9	55.5±6.9^{1}	81.7±5.4^{1}	80.2±5.5^{5}	76.6±5.7^{3}
PT	40.3±4.8	24.3±3.4^{1}	31.4±4.1^{1}	41.4±5.3^{5}	38.9±4.9^{4}
SO	100.0±0.0	25.9±16.2^{1}	100.0±0.0^{6}	97.3±4.9^{1}	97.4±3.1^{1}
VO	95.2±1.5	60.7±3.0^{1}	94.6±1.3^{3}	95.5±1.5^{5}	95.4±1.5^{6}
WI	96.9±2.0	36.9±8.6^{1}	96.5±2.1^{5}	90.5±5.5^{1}	93.0±4.2^{1}
AD	77.0±5.3	20.8±3.6^{1}	75.8±5.2^{5}	63.0±6.6^{1}	66.9±6.0^{1}
AN	97.4±0.9	76.2±2.2^{1}	98.8±0.8^{1}	92.9±2.1^{1}	93.4±1.7^{1}
EC	64.6±7.0	68.1±6.9^{1}	63.0±5.4^{5}	67.9±6.0^{1}	65.0±6.4^{6}
GL	70.6±5.8	31.4±5.0^{1}	66.7±6.2^{1}	55.0±6.9^{1}	66.4±7.4^{1}
HD	79.7±3.8	55.1±3.8^{1}	78.8±3.8^{5}	78.5±3.9^{5}	77.3±3.5^{1}
HO	82.6±3.9	64.1±3.8^{1}	76.1±4.0^{1}	82.2±3.6^{6}	81.0±3.8^{2}
HY	97.5±0.4	95.3±0.5^{1}	97.3±0.5^{1}	98.3±0.5^{1}	99.2±0.2^{1}
LI	59.9±7.2	7.3±3.2^{1}	53.0±7.0^{1}	57.5±7.7^{2}	57.1±8.1^{2}
MU	100.0±0.2	51.8±0.9^{1}	100.0±0.0^{6}	100.0±0.0^{6}	100.0±0.0^{6}
PO	64.1±7.0	71.0±6.2^{1}	58.3±7.9^{1}	59.4±7.7^{1}	67.3±9.7^{3}
PR	86.8±5.2	42.1±5.3^{1}	90.6±5.2^{1}	72.0±8.9^{1}	80.6±8.5^{1}
SF	71.6±3.7	25.4±4.1^{1}	68.2±3.5^{1}	70.7±3.4^{4}	71.5±3.7^{6}
SN	77.9±9.3	51.0±7.0^{1}	75.0±5.6^{3}	63.0±5.5^{1}	69.6±8.0^{1}
SP	93.1±1.6	52.2±1.7^{1}	94.3±0.6^{1}	90.9±1.0^{1}	93.2±1.1^{6}
ZO	93.9±3.7	40.5±6.3^{1}	94.9±4.2^{3}	92.4±5.5^{3}	91.2±5.1^{1}

Table 4.3. Summary of accuracy results.

	Test datasets				All datasets			
Measure	RISE	PEBLS	CN2	C4.5	RISE	PEBLS	CN2	C4.5
No. wins	-	10-4	12-2	10-4	-	20-8	20-9	18-11
No. signif. wins	-	7-4	10-2	9-2	-	14-7	18-6	15-7
Wilcoxon test	-	96.0	99.6	98.0	-	98.0	99.6	98.0
Average	81.1	79.4	76.2	78.6	79.7	78.3	76.4	77.8
Score	3.1	2.3	1.8	2.2	2.8	2.2	2.1	2.4

taken from the set of real-world datasets, consider the accuracy difference in each dataset as a sample of a random variable, and test the hypothesis of whether this difference is in general positive. The Wilcoxon signed-ranks test is a procedure that addresses this issue (DeGroot, 1986). It takes into account the signs of the observed differences, and their ranking (i.e., the largest difference counts more than the second largest one and so forth, but it does not matter by how much). In each case, the results support the hypothesis that RISE is a more accurate algorithm with a confidence in excess of 95%. It is therefore highly unlikely that RISE performed better than the other algorithms by chance. If the domains used constitute a good sample of the set of real-world domains to which these algorithms will be applied, we can then conclude with high confidence that RISE is the most accurate one.

It should be remarked that, unlike what is common practice in the machine learning literature, this test is being performed at the meta level: the question being asked is "Is RISE better than the other algorithm on this ensemble of datasets?", as opposed to asking 30 times "Is RISE better than the other algorithm on an ensemble of test sets from the same dataset?" Thus the use of a large number of datasets does not undermine the conclusions reached, but instead makes the very high confidences obtained possible.

- *Average.* The average performance across all datasets is a measure of debatable significance, but it is often reported (e.g., (Quinlan, 1993b; Clark & Boswell, 1991)) and provides additional perspective. Again RISE does visibly better than every other algorithm. It is also interesting to note that, although IBL (PEBLS) and rule induction (CN2) often differ by large margins in specific datasets, globally these differences tend to cancel each other out. This is also true when comparing C4.5 with PEBLS.
- *Score.* The score is a measure that compares all five algorithms simultaneously (default included), looking not only at which one is the most accurate, but also taking into consideration that being the second is better than being the worst, etc. Specifically, for each dataset the most accurate algorithm receives 4 points, the second 3, the third 2, the fourth 1 and the worst 0. An

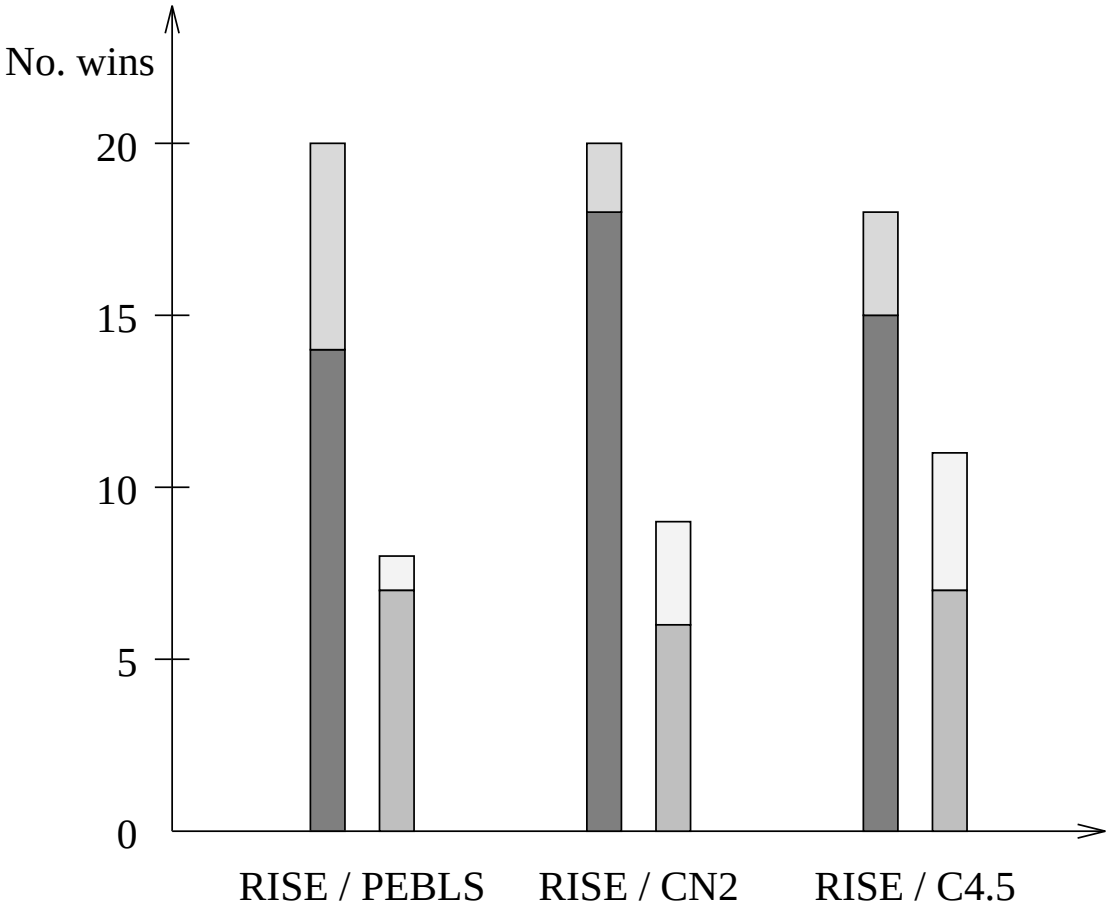

Figure 4.1. Comparison of accuracies: RISE vs. PEBLS, CN2 and C4.5.

algorithm's score is then defined as the average number of points it received (i.e., the sum of the points received, divided by the number of datasets). RISE obtains the largest score by a wide margin.

The results for all datasets are also shown in bar graph form in Figure 4.1, where each pair of bars compares RISE with one of the other systems. The left-hand bar in each pair represents the number of RISE wins vs. the other system, and the right-hand bar represents the number of the other system's wins. The darker portion of each bar represents the number of significant wins at the 95% confidence level.

The same general pattern of RISE wins typically emerges if the results are broken down by dataset size (small, medium and large, i.e., $e < 100$, $100 \leq e < 1000$, and $e \geq 1000$), dataset type (symbolic, numeric and mixed), difficulty (easier and harder, as measured by the highest accuracy being above or below 75%, the global average), and application area (medical diagnosis, engineering, social science, life sciences and miscellaneous). Confidence levels are of course reduced due to the smaller sample size, and there is sometimes no clear pattern

when the number of datasets is small, but RISE's excellent results clearly hold across the board.

A very significant observation is that in 14 datasets RISE's accuracy exceeds the highest of CN2's and PEBLS's (i.e., RISE not only matches the results of the best of its parent approaches, but is able to improve on them). In nine of those datasets this is true with a confidence level of 95% or higher, using a one-tailed paired t test (see Table 4.2). This shows that a significant synergy can be obtained by unifying multiple empirical learning approaches.

RISE was also compared with the CART decision tree learner (Breiman, Friedman, Olshen & Stone, 1984) (in the IND implementation (Buntine & Caruana, 1992)) on the datasets above and others, leading to similar results. For example, in the StatLog letter recognition dataset (Michie, Spiegelhalter & Taylor, 1994) RISE obtained an average accuracy of 93.2%, vs. CART's 83.8%.

4.6 Lesion Studies

The empirical results just described lead to the conclusion that RISE represents a significant advance in the state-of-the-art in empirical concept learning. However, we would like to verify that RISE's improved performance is indeed due to its unification of rule induction and IBL, and, in general, to know what factors RISE's higher accuracy should be attributed to. To answer this, lesion studies were conducted, and additional aspects of the algorithm's performance were measured. Lesion study results are reported in Table 4.4. The first column shows the full system's accuracy, and each of the following ones represents a lesioned version. Superscripts indicate confidence levels for the accuracy differences between systems, again using a one-tailed paired t test. These results are summarized in Table 4.5, and in bar graph form in Figure 4.2, using the same representation as before. The results of performance monitoring for the full RISE system are shown in Table 4.6. The first four columns show, respectively: the percentage of test cases that were not matched by any rule, but had a single closest rule, or for which all equally close rules were of the same class (No/One); the percentage not matched by any rule, and for which there were equally close rules of more than one class (No/Mlt); the percentage matched by only one rule, or rules of only one class (One); and the percentage matched by rules of more than one class (Mlt). The next four columns show RISE's average accuracy given each of these situations (e.g., column five shows RISE's average accuracy on test examples that were not matched by any rule, but had a single closest rule/class). The bottom line of the table shows the averages of the corresponding columns over all datasets. These observations aid in interpreting the lesion study results.

The first specific question addressed was whether there is any gain in the rule induction process (i.e., whether RISE constitutes an improvement over pure

Table 4.4: Empirical results: lesion studies. Superscripts denote confidence levels: 1 is 99.5%, 2 is 99%, 3 is 97.5%, 4 is 95%, 5 is 90%, and 6 is below 90%.

Dataset	RISE	IBL	Rules	No t-b.
BC	67.7	65.1^{1}	69.0^{4}	68.7^{1}
CE	83.3	81.3^{1}	66.9^{1}	83.2^{6}
CH	98.2	91.9^{1}	91.9^{1}	98.0^{1}
DI	70.4	70.3^{6}	66.2^{1}	70.5^{1}
HE	78.3	78.4^{6}	79.6^{5}	78.5^{5}
IR	94.0	94.7^{3}	71.0^{1}	94.0^{6}
LA	87.2	90.8^{1}	73.7^{1}	87.1^{6}
LC	44.7	42.0^{6}	26.5^{1}	44.2^{5}
LD	62.4	60.9^{4}	62.1^{6}	62.3^{6}
LE	77.2	72.5^{1}	64.8^{1}	75.8^{3}
LY	78.7	82.0^{1}	70.1^{1}	78.2^{3}
PT	40.3	34.3^{1}	33.5^{1}	37.0^{1}
SO	100.0	100.0^{6}	85.1^{1}	100.0^{6}
VO	95.2	94.6^{3}	83.7^{1}	94.2^{1}
WI	96.9	95.1^{1}	51.5^{1}	96.9^{6}
AD	77.0	75.8^{5}	55.1^{1}	76.2^{1}
AN	97.4	97.7^{4}	77.7^{1}	97.2^{1}
EC	64.6	59.2^{1}	65.7^{6}	64.5^{6}
GL	70.6	68.3^{2}	47.3^{1}	70.4^{2}
HD	79.7	77.8^{1}	64.7^{1}	79.7^{6}
HO	82.6	76.6^{1}	79.0^{1}	81.7^{1}
HY	97.5	94.1^{4}	84.8^{1}	97.5^{6}
LI	59.9	55.9^{1}	52.7^{1}	49.7^{1}
MU	100.0	97.5^{6}	100.0^{6}	99.8^{1}
PO	64.1	59.1^{1}	70.9^{1}	65.9^{1}
PR	86.8	90.6^{1}	68.4^{1}	85.9^{4}
SF	71.6	71.1^{6}	65.5^{1}	68.1^{1}
SN	77.9	83.8^{1}	52.9^{1}	77.9^{6}
SP	93.1	87.8^{1}	75.9^{1}	92.1^{1}
ZO	93.9	94.5^{4}	81.0^{1}	93.7^{4}

Table 4.5. Summary of lesion study results.

	Test datasets				All datasets			
Measure	RISE	IBL	Rules	No t.-b.	RISE	IBL	Rules	No t.-b.
No. wins	-	11-4	12-2	11-1	-	21-8	25-4	20-4
No. sigf. wins	-	8-4	12-1	10-1	-	16-7	24-2	15-3
Wilcoxon test	-	97.0	99.8	99.0	-	99.5	99.9	99.8
Average	81.1	79.3	69.4	80.0	79.7	78.1	67.9	79.0

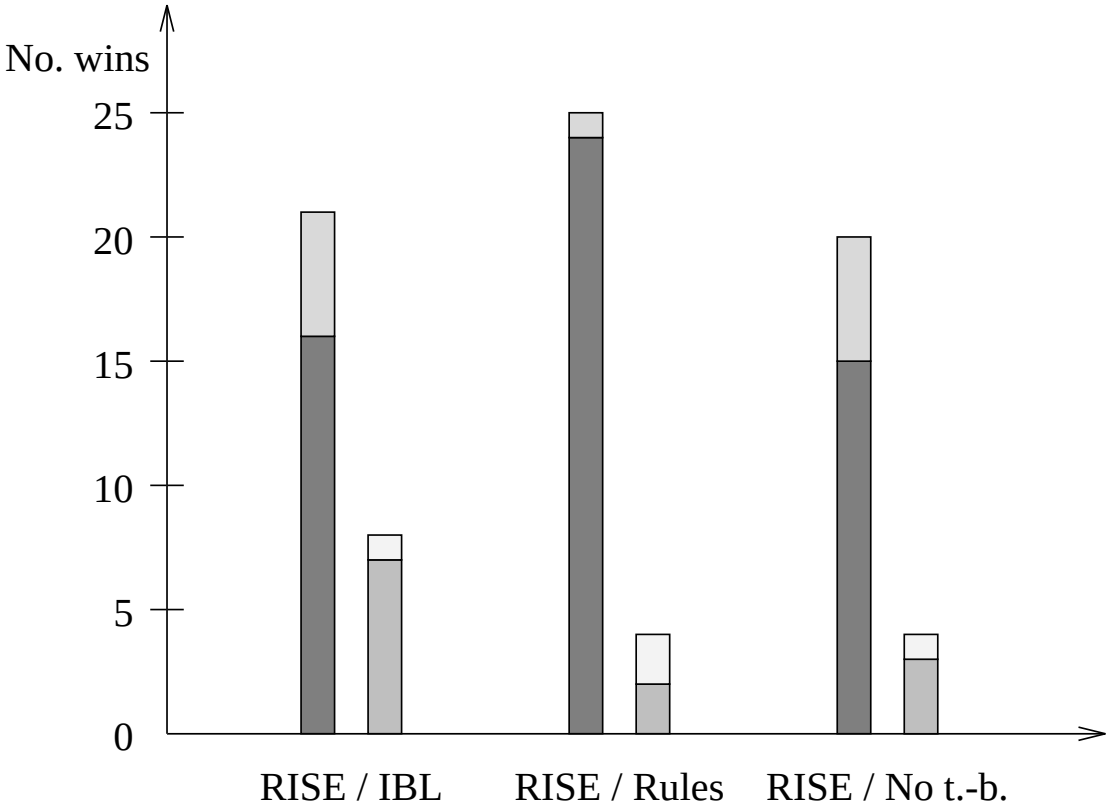

Figure 4.2: Comparison of RISE with its lesioned versions: IBL only, rule induction only, and no tie-breaking.

Table 4.6. Empirical results: performance monitoring.

	Match type frequency				Accuracy per match type			
Dataset	No/One	No/Mlt	One	Mlt	No/One	No/Mlt	One	Mlt
BC	33.4	1.5	59.1	6.0	59.7	58.6	73.5	56.1
CE	51.9	0.0	46.9	1.1	79.3	25.0	88.1	65.6
CH	27.6	0.1	71.4	0.9	95.5	65.0	99.5	87.4
DI	70.9	0.1	27.4	1.6	67.7	62.5	78.0	58.5
HE	55.3	0.1	43.1	1.5	70.9	33.3	88.7	53.8
IR	45.9	0.0	54.0	0.2	89.6	–	97.9	50.0
LA	51.8	0.2	46.6	1.4	81.9	0.0	95.0	30.8
LC	88.2	0.4	9.1	2.4	49.4	0.0	30.0	53.8
LD	72.7	0.2	24.0	3.0	59.5	50.0	72.6	54.1
LE	13.2	1.5	83.0	2.2	45.3	16.7	83.4	77.8
LY	42.9	0.4	53.9	2.8	66.3	55.6	89.4	65.2
PT	34.2	4.1	41.6	20.1	34.8	19.7	48.7	36.2
SO	16.9	0.0	83.1	0.0	100.0	–	100.0	–
VO	7.4	0.1	90.3	2.1	77.8	33.3	97.4	65.6
WI	78.1	0.0	21.9	0.0	96.1	–	100.0	–
AD	53.6	1.6	43.0	1.8	62.8	37.0	97.3	65.5
AN	24.1	0.0	75.6	0.2	91.7	50.0	99.8	78.8
EC	70.1	0.1	26.8	3.0	63.9	100.0	66.2	66.7
GL	71.5	0.0	27.2	1.3	65.5	–	84.3	67.4
HD	63.1	0.0	34.8	2.1	77.0	0.0	85.7	65.4
HO	39.7	0.2	55.4	4.6	72.5	81.8	89.8	83.5
HY	40.7	0.1	58.3	0.9	95.8	66.7	99.0	81.4
LI	14.7	4.5	47.3	33.5	34.7	32.4	76.2	51.6
MU	7.3	0.0	92.5	0.2	99.9	–	100.0	100.0
PO	36.5	12.2	44.7	6.6	59.3	76.1	63.4	76.8
PR	59.7	0.0	35.8	4.5	85.2	–	91.4	71.8
SF	16.7	2.9	59.9	20.4	59.2	55.8	81.8	54.0
SN	95.6	0.0	4.3	0.1	77.4	–	89.9	0.0
SP	67.4	0.0	29.9	2.7	92.6	50.0	95.8	75.2
ZO	17.5	0.0	81.9	0.5	74.8	–	98.3	44.4
Average	45.6	1.1	49.1	4.3	72.9	44.1	85.4	62.1

instance-based learning). The "IBL" column in Table 4.4 reports the accuracies obtained by the initial, ungeneralized instance set, and shows that generalization often produces significant gains in accuracy, while seldom having a negative effect. Not surprisingly, the results are similar to those of RISE vs. PEBLS.

The converse question is whether the instance-based component is really necessary. Simply assigning examples not covered by any rule to a default class, as done in most rule induction systems, might be sufficient. The "Rules" column in Table 4.4 shows the results obtained using this policy, and confirms the importance of "nearest-rule" classification in RISE. The sum of the "No" columns in the left half of Table 4.6 is the percentage of test cases assigned to the default class. This is often very high, the more so because RISE tends to produce rules that are more specific than those output by general-to-specific inducers. The use of nearest-rule is thus essential. Note that the results reported in the "Rules" column are for applying the default rule during both learning and classification; applying it exclusively during classification produced only a slight improvement.

Another important component of RISE whose utility needs to be determined is the conflict resolution policy, which in RISE consists of letting the tied rule with the highest Laplace accuracy win. This was compared with simply letting the most frequent class win ("No t-b." column in Table 4.4). The sum of the "Mlt" columns in the left half of Table 4.6 is the percentage of cases where tie-breaking is necessary. This is typically small, and the increase in accuracy afforded by RISE's conflict resolution strategy is correspondingly small (0.7% on average, for all datasets). However, this increase is consistently produced, as evinced by the fact that RISE is more accurate than its lesioned version with a 99.8% confidence by the Wilcoxon test.

Taken together, the lesion studies show that each of RISE's components is essential to its performance, and that it is their unification in one system that is responsible for the excellent results obtained by RISE *vis-à-vis* other approaches.

The question now becomes one of whether RISE's unified approach is superior to the individual ones for the reasons hypothesized in the previous chapters, and, in general, under what conditions the unified approach will lead to superior performance. This was investigated by conducting experiments with artificial datasets. These studies are described in the next two chapters.

The performance monitoring results (Table 4.6) are also of interest in themselves. For example, they show that RISE is more accurate when a single rule/class wins the example (No/One and One) than when conflict resolution is necessary (No/Mlt and Mlt), and that RISE is more accurate when rules cover the example (One and Mlt) than when they are at some nonzero distance from it (No/One and No/Mlt), with the multiplicity effect being stronger than the distance effect. Thus, Acc(One) > Acc(No/One) > Acc(Mult) > Acc(No/Mlt), forming a natural progression from the less ambiguous cases to the more ambiguous ones. Intuitively,

when rules of a single class match the example there is more confidence in the resulting class prediction than when conflict resolution is necessary, and when there is a perfect match between the winning rule and the example there is more confidence in the prediction than when the match is only approximate. Comparing the number of domains in which each match type is more accurate than each other leads to the same conclusions. Finally, comparing the two halves of Table 4.6 shows that RISE is more accurate in the situations that occur most often, which is the most desirable outcome with respect to the goal of maximizing global accuracy.

4.7 Space and Time Comparisons

Besides accuracy, two variables of interest are output size and running time. These are listed in Tables 4.7 and 4.8. Output size is measured by counting one unit for each antecedent of a rule, plus one for the consequent. The size for PEBLS is simply that of the the training set (i.e., $e(a+1)$, in the notation of Section 3.5). Running times were obtained on a Sun 670 machine, with all algorithms implemented in C and similarly compiled. All values shown are averages of 50 runs.

With respect to output size, RISE occupies an intermediate position between the IBL and rule induction representatives. It obtains large savings compared to PEBLS, and these increase with training set size (see, e.g., the MU dataset). We can thus expect RISE to be at a significant advantage relative to unedited IBL methods in applications where a large number of examples is available, and the target concept is comparatively simple. On the other hand, RISE's output is still generally much larger than CN2's. The number of rules produced (not shown) is typically similar, but RISE's rules tend to be substantially more specific; this is consistent with the search direction and best-match policy it uses. An important fact is that, if rules that do not win any training examples are discarded, the resulting decrease in RISE's accuracy is minimal (0.05% on average) and has practically no effect on the comparison with CN2, but the output size is reduced by 50% or more, and the average number of rules becomes smaller than CN2's in every dataset (except VO, where it is about the same). Thus, when simplicity of the output is a goal, RISE can often come quite close to CN2 without compromising accuracy. However, the simplest results are always those obtained by C4.5RULES. (The pruned tree sizes (not shown) are comparable to those of CN2's and RISE's pruned rule sets.)

With respect to running time, RISE is typically slower than PEBLS. This is to be expected, since RISE essentially does everything that PEBLS does, and more. More surprising is the fact that RISE is still faster than PEBLS on eight datasets, including several purely symbolic ones. This suggests that PEBLS's implementation could be further optimized. RISE is faster than CN2 on most datasets, but this advantage tends to be concentrated in the smaller ones, raising the possibility that CN2 will be at a considerable advantage in larger datasets than

Table 4.7. Empirical results: average output sizes.

Dataset	RISE	PEBLS	CN2	C4.5
BC	1045.4	1910	410.2	17.5
CE	4110.9	7392	679.4	51.0
CH	3063.8	79217	843.1	96.2
DI	3048.8	4626	802.0	69.3
HE	1231.9	2060	112.8	20.5
IR	382.9	500	63.2	9.9
LA	305.7	646	42.3	10.0
LC	403.3	1197	39.6	7.8
LD	1156.8	1617	436.1	63.6
LE	16.7	80	25.5	6.1
LY	756.4	1881	131.7	25.4
PT	2076.7	4086	653.6	61.1
SO	106.2	1116	18.8	9.9
VO	541.0	4947	113.2	15.9
WI	896.3	1666	87.6	14.3
AD	4165.8	9380	236.4	40.7
AN	13527.3	20826	420.4	62.0
EC	550.2	696	137.7	19.7
GL	696.2	1430	245.8	47.0
HD	1460.5	2842	320.8	42.9
HO	2111.6	4623	268.6	19.4
HY	7465.9	55094	430.7	14.5
LI	217.3	536	125.8	40.3
MU	398.1	125189	176.8	38.6
PO	470.3	540	132.9	4.5
PR	1198.6	4118	82.5	16.0
SF	1316.2	2808	326.3	35.2
SN	3756.4	8479	224.0	33.9
SP	8350.7	130357	1360.5	211.3
ZO	196.2	1139	41.3	20.0

Table 4.8: Empirical results: average running times (in minutes and seconds), and ratio of running times of PEBLS, CN2 and C4.5 to running time of RISE (PB/R, CN/R and C4/R, respectively).

Dataset	RISE	PEBLS	CN2	C4.5	PB/R	CN/R	C4/R
BC	0:15.11	0:03.16	0:22.86	1:31.41	0.2	1.5	6.0
CE	4:31.25	0:16.11	2:05.30	2:53.67	0.1	0.5	0.6
CH	10:39.67	11:59.41	6:41.85	4:00.51	1.1	0.6	0.4
DI	4:15.37	0:12.30	2:13.93	4:02.76	0.0	0.5	1.0
HE	0:10.93	0:02.32	0:07.42	0:08.24	0.2	0.7	0.8
IR	0:05.42	0:01.74	0:03.01	0:02.09	0.3	0.6	0.4
LA	0:01.30	0:01.60	0:02.42	0:01.96	1.2	1.9	1.5
LC	0:00.69	0:01.70	0:04.64	0:02.07	2.5	6.7	3.0
LD	1:31.02	0:02.82	0:28.21	1:06.12	0.0	0.3	0.7
LE	0:00.13	0:01.78	0:00.55	0:01.31	13.7	4.2	10.1
LY	0:04.88	0:02.21	0:07.77	0:06.07	0.5	1.6	1.2
PT	0:14.38	0:06.07	1:58.39	1:55.04	0.4	8.2	8.0
SO	0:00.50	0:01.64	0:01.46	0:01.74	3.3	2.9	3.5
VO	0:27.71	0:07.07	0:06.47	0:03.41	0.3	0.2	0.1
WI	0:15.31	0:02.77	0:10.30	0:11.18	0.2	0.7	0.7
AD	0:11.20	0:07.14	1:10.47	0:24.18	0.6	6.3	2.2
AN	4:26.07	1:46.58	4:34.74	2:46.27	0.4	1.0	0.6
EC	0:04.84	0:02.70	0:07.16	0:13.05	0.6	1.5	2.7
GL	0:11.03	0:02.75	0:28.22	1:41.71	0.2	2.6	9.2
HD	0:26.35	0:03.71	0:24.85	1:53.25	0.1	0.9	4.3
HO	1:39.64	0:05.77	1:42.63	0:24.29	0.1	1.0	0.2
HY	14:45.96	8:36.18	11:53.69	1:54.54	0.6	0.8	0.1
LI	0:00.60	0:01.68	0:02.42	0:02.89	2.8	4.0	4.8
MU	10:07.22	45:37.28	4:16.37	18:29.79	4.5	0.4	1.8
PO	0:01.49	0:01.46	0:05.28	0:02.53	1.0	3.5	1.7
PR	0:07.85	0:02.94	0:15.03	0:06.28	0.4	1.9	0.8
SF	0:08.97	0:04.19	0:20.03	0:28.49	0.5	2.2	3.2
SN	2:35.81	0:09.76	2:39.08	5:15.54	0.1	1.0	2.0
SP	51:28.15	19:25.15	32:15.24	24:20.94	0.4	0.6	0.5
ZO	0:01.11	0:01.76	0:02.03	0:02.16	1.6	1.8	1.9

those used here, provided the concept description is still sufficiently simple (if not, CN2's time will grow to the worst-case bound discussed in the previous chapter, equivalent to RISE's). RISE is also typically faster than C4.5 in this comparison, but this is due to the fact that the windowing option is being used in C4.5; even though each tree is typically grown faster with windowing, this gain tends to be swamped by the fact that 10 trees are being grown, and rules are being selected from all of them.

We conclude that, for datasets of very large size, and/or when comprehensibility is paramount, a system like C4.5 will still be the first choice.[4] If, on the other hand, accuracy is the dominant goal, the results of previous sections indicate that RISE may be a more appropriate algorithm to use.

4.8 Summary

The unified approach to concept learning embodied in RISE has been validated by extensive empirical testing. RISE is significantly more accurate than state-of-the-art representatives of the approaches it unifies (PEBLS and CN2) in a large number of application datasets, and often more accurate than the best of the two. RISE also significantly outperforms an advanced decision-tree learner (C4.5) on most datasets used. Lesion studies show that each of RISE's aspects is essential to its performance; neither by itself would achieve it. RISE's output size is typically larger than CN2's and smaller than PEBLS's. Running times are comparable for all systems, with RISE being typically somewhat slower than PEBLS, and faster than CN2.

[4] Assuming output size is a reasonable measure of comprehensibility.

Chapter 5
RISE as Rule Induction

5.1 Overview

This chapter investigates empirically how RISE's behavior differs from that of other rule induction algorithms, and extends RISE with methods for combining its output with the output of those algorithms. Studies in artificial domains show the use of RISE to be particularly advantageous when the concepts to be learned are fairly to highly specific. The combination of RISE with general-to-specific rule learners, resulting in *two-way induction*, is observed empirically to lead to further gains in accuracy.

5.2 Experiments in Artificial Domains

Compared to "divide and conquer" or "separate and conquer" systems like C4.5 and CN2, RISE's higher accuracies can be hypothesized to be due to several factors:

- RISE's "conquering without separating" induction strategy makes it less sensitive to the fragmentation problem.
- RISE's specific-to-general search direction allows it to identify small exception regions that may go undetected (or be more poorly identified) by general-to-specific learners.
- RISE is able to form complex non-axis-parallel frontiers, and is thus at an advantage when these are appropriate.

RISE's advantages should be particularly apparent when attempting to learn concepts that are fairly to highly specific. These are concepts that cover only small, fragmented regions of the instance space (semantic specificity), and/or whose descriptions include a large fraction of the attributes used to describe examples (syntactic specificity). If a region whose frontiers are non-axis-parallel contains few examples, approximating those frontiers should be harder for axis-parallel learners than if the region contains many examples, since, on average, a region containing fewer examples will contain proportionally fewer examples close to the frontier,

where they are critical for correct induction. Concepts of significant specificity should also be more strongly affected by the fragmentation problem than more general ones, since they will encounter earlier on the lack of examples produced by fragmentation. Finally, as in any search process, finding the target tends to be harder when it lies farther from the search's starting point, which in the case of general-to-specific learners like C4.5 and CN2 is the null concept description. When this happens, the opportunities for the search to take the wrong path accumulate, and the probability of an incorrect end result increases. As a result, if the concept description is fairly specific, systems like C4.5 and CN2 may have trouble finding it in concept space. A specific-to-general learner like RISE, on the other hand, should have less difficulty inducing fairly specific descriptions, since it starts from specific examples and is thus closer to the target. By the same logic, however, it can be at a disadvantage when the target concepts are general: since search has to start from specific examples, noise in these can more easily interfere with learning general rules; early decisions, based on little evidence, can be wrong and difficult to correct later on.

This section describes a test of the hypothesis that RISE will be more accurate than a "divide and conquer" algorithm when the target concepts are fairly to very specific, with the advantage increasing with specificity. The independent variable of interest is thus the specificity of the target concept description. A good operational measure of it is the average length of the rules comprising the correct description: rules with more conditions imply a syntactically more specific concept; additionally, if the distribution of examples in instance space is approximately uniform, they will also imply a semantically more specific concept. The dependent variables are the out-of-sample accuracies of RISE and "divide and conquer" algorithms; C4.5RULES (Quinlan, 1993a) was chosen as a representative of these. Concepts defined as Boolean functions in disjunctive normal form were used as targets. The datasets were composed of 100 examples described by 16 attributes. The average number of literals C in each disjunct comprising the concept was varied from 1 to 16. The number of disjuncts was set to $\mathrm{Min}\{2^{C-1}, 25\}$. This attempts to keep the fraction of the instance space covered by the concept roughly constant, up to the point where it would require more rules than could possibly be learned. Equal numbers of positive and negative examples were included in the dataset, and positive examples were divided evenly among disjuncts. Thus, until the maximum of 25 disjuncts is reached, the distribution of examples in the instance space is approximately uniform.[1] In each run a different target concept was used, generating the disjuncts at random, with length given by a binomial distribution with mean C and variance $C(1 - \frac{C}{16})$; this is obtained by including each feature in the disjunct with probability $\frac{C}{16}$. Twenty runs were conducted, with two-thirds of the data used for training and the remainder for testing.

The results are shown graphically in Fig. 5.1. The most salient aspect is the large difference in difficulty between short and long rules for both learners.

[1] Approximately, and not exactly, because there may be overlap between disjuncts.

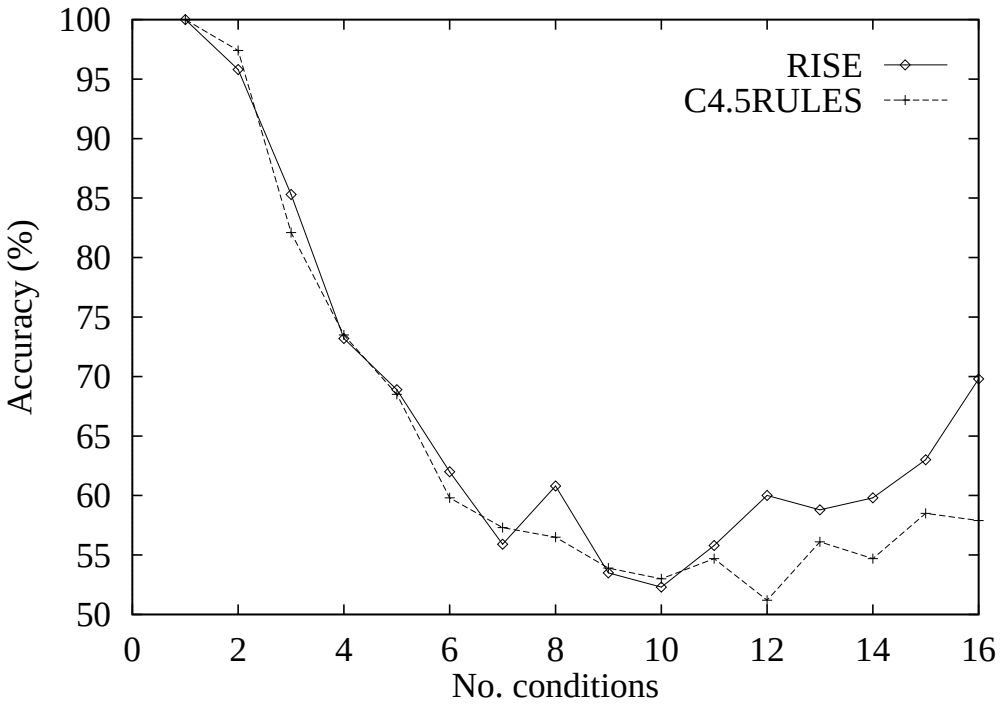

Figure 5.1. Accuracy as a function of concept specificity (16 features).

Concepts with very few (approx. three or less) conditions per rule are so simple that both RISE and C4.5RULES are able to learn them easily. In separate experiments, corrupting the data with 10% and 20% class noise degraded the performance of the two algorithms equally, again giving no advantage to C4.5RULES. At the other end, however, RISE has a clear advantage for concepts with 12 or more conditions per rule; all differences here are significant with 95% confidence using a one-tailed paired t test.[2]

The slight upward trend in C4.5RULES's curve for $C > 10$ was investigated by repeating the experiments with 32 attributes, 400 examples, a maximum of 50 rules and $C = 1, \ldots, 32$. The results are shown in Fig. 5.2. C4.5RULES's lag increases, but the upward trend is maintained; on inspection of the rules C4.5RULES produces, this is revealed to be due to the fact that, as the concept rules become more and more specific, it becomes possible to induce short rules for its negation. The hardest concepts, for which both the concept and its negation have necessarily long rules, are for intermediate values of C.

[2]These confidence levels apply to the accuracy difference in the entire domain class studied, not just a particular dataset, since the training sets were drawn independently from the domain class.

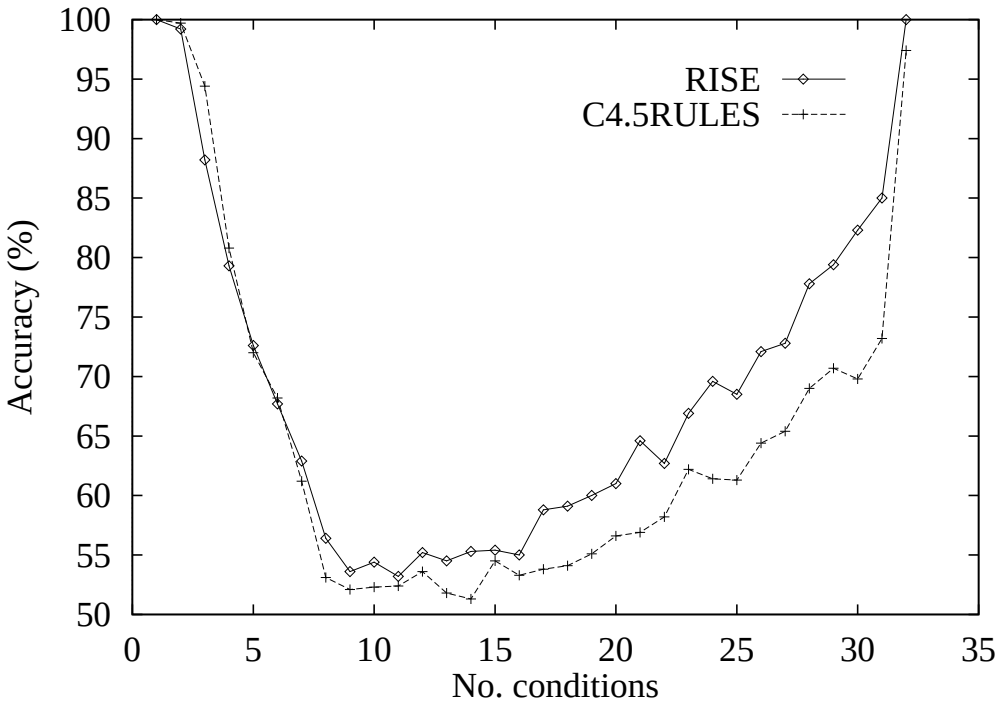

Figure 5.2. Accuracy as a function of concept specificity (32 features).

In summary, the results of this experiment support the hypothesis that the specificity of the regions to be learned is a factor in the difference in accuracy between RISE and "divide and conquer" rule induction systems, with greater specificity favoring RISE.

5.3 An Extension of RISE: Two-Way Induction

In practice, most concepts appear to be best described, not by general rules alone or by specific ones alone, but by a combination of the two. In datasets from the UCI repository, RISE, CN2 and C4.5RULES all typically produce such a mixture of rules. When both specific and general rules are needed to describe a concept, both general-to-specific and specific-to-general approaches can be expected to have problems, since the presence of rules of one type is likely to interfere with learning rules of the other. General-to-specific learners may not recognize the exception areas, and specific-to-general ones may induce only imperfect, corrupted versions of the general rules. A natural solution would then be to combine the two search directions in a single *two-way induction* system, allowing both the induction of rules starting from the null description and from specific examples, and employing some conflict resolution strategy in parts of the instance space where

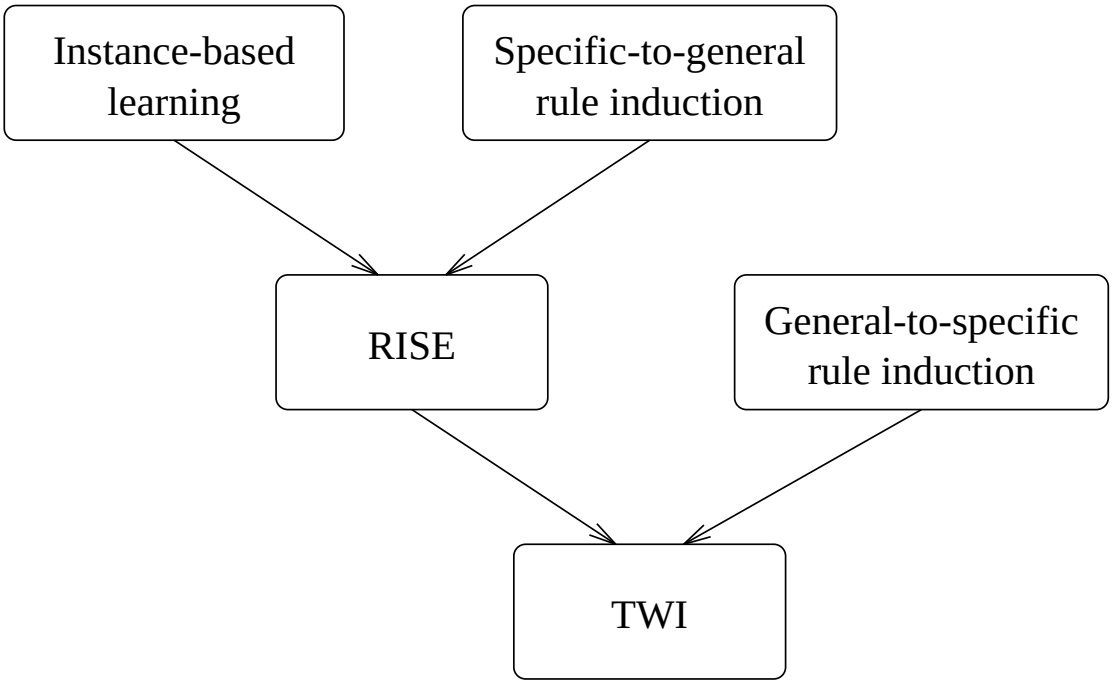

Figure 5.3. Two-way induction using RISE.

rules' predictions disagree. This section describes and evaluates one realization of this idea, using RISE as the specific-to-general learner, and C4.5RULES and CN2 in turn as the general-to-specific one. The resulting multistrategy learner, called TWI, is shown graphically in Figure 5.3.

Applying the RISE algorithm to a training set produces a set of rules which will henceforth be called the S rules (for "specific"). Applying the general-to-specific learner produces a set of G rules. Two algorithms for combining the S and G rules were designed, TWI-Y and TWI-U. In TWI-Y, the sets of S and G rules are merged to form a single set, deleting any duplicates. The Laplace accuracy of each rule on the examples that it *covers* is then measured, and the classification procedure is applied to each training example in turn; each rule memorizes the number of examples it won, and how many of them it classified correctly. At the end, the Laplace accuracy of each rule on the examples that it *won* is computed, and this is the value retained. The earlier estimates were necessary to break ties during the classification cycle that produced the final ones. At classification time, the procedure described in the previous section is applied without any distinction between S and G rules. The nearest rule to the test example wins; if two or more rules are equally near, the one with the highest Laplace accuracy prevails. This algorithm, TWI-Y, is shown graphically in Figure 5.4, where the Y structure that gives it its name is apparent.

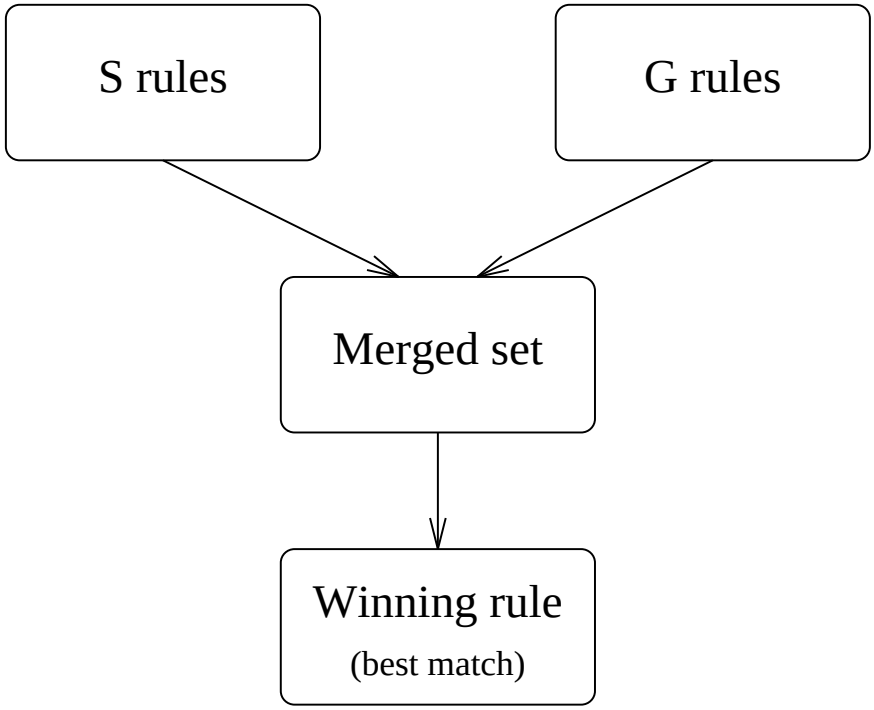

Figure 5.4. The TWI-Y algorithm.

In the TWI-U algorithm, the sets of S and G rules are kept separate, and the Laplace accuracy of each rule on the training examples that it covers is measured. At classification time, the two sets of rules are first applied separately. A winner among the S rules is found by the procedure previously outlined. To select the G winner, the G rules are matched against the test example. If only one G rule covers it, that rule is chosen as the G winner. If more than one rule covers the example, the one with the highest Laplace accuracy wins. If no G rule covers the example, the default rule is chosen as the G winner. Of the two finalists (S and G), the one with the highest accuracy then wins and classifies the example. This algorithm is shown graphically in Figure 5.5.

An empirical study was conducted to evaluate the performance of the two-way induction system relative to its individual components. The default settings for C4.5, C4.5RULES and CN2 were kept. All results for C4.5RULES (abbreviated C4.5R) and CN2 below are those obtained using their own classification procedures, not the one above for G rules. Twenty-four datasets from the UCI repository were used (see Section 4.2), and 20 runs were conducted for each dataset, with two-thirds of the data used for training. The default classifier (assigning examples to the most frequent class) was included as a baseline. The results are shown in Table 5.1 for C4.5RULES used as the G component, and in Table 5.2 for CN2. The average accuracy and sample standard deviation are tabulated for each algorithm in each domain. The superscripts denote confidence levels for the difference in accuracy

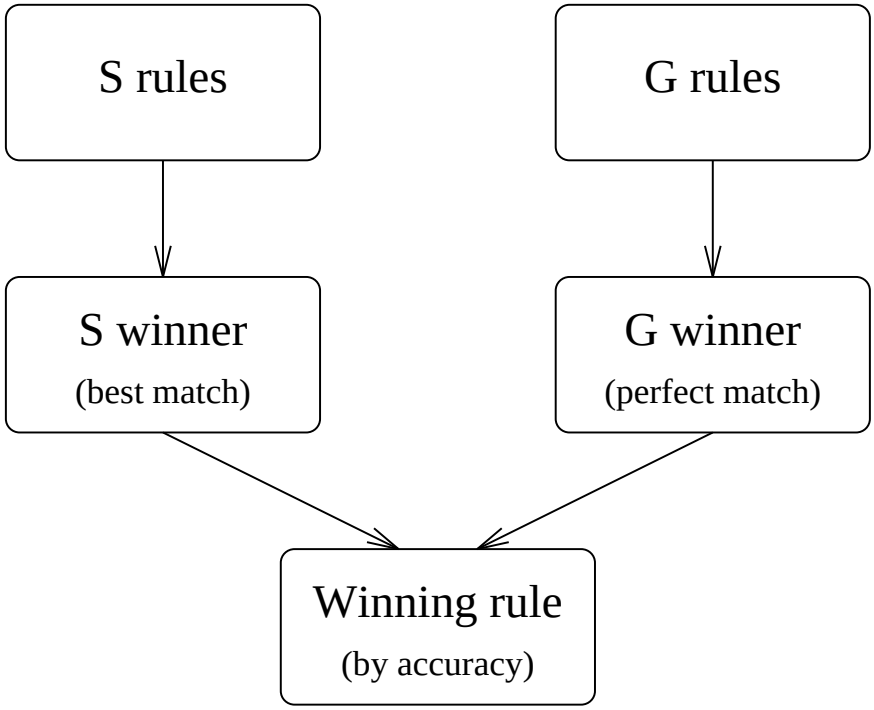

Figure 5.5. The TWI-U algorithm.

between TWI-U and the corresponding algorithm, using a one-tailed paired t test. For example, in the horse colic domain in Table 5.1, TWI-U obtains an accuracy of 83.8% with a standard deviation of 3.5%, is better than C4.5RULES with 99% confidence, and better than RISE with 99.5% confidence.

Table 5.3 summarizes these results. The first line shows the number of domains in which TWI-U achieved higher accuracy than the corresponding system, vs. the number in which the reverse happened. The second line considers only those domains in which the observed difference is significant with at least 95% confidence, and shows that most of the previous "wins" were indeed significant. For example, when G was C4.5RULES, TWI-U did better than C4.5RULES in 17 domains overall, and worse in 7; with 95% confidence it did better in 12 and worse in 2. All other comparisons are similarly favorable to TWI-U. The third line shows the results of applying a sign test to the values of line one. This consists of considering the number of wins as a binomial variable, and asking how unlikely the value obtained is if TWI-U is assumed to be no more accurate than the corresponding algorithm. For example, 17 wins in 24 trials has a probability of occurrence of only 3%. This test shows that all the numbers of wins obtained are significant at the 95% confidence level, resulting in high certainty that TWI-U is a more accurate learner than either component alone, in the set of learning tasks of which the domains used constitute a sample.

Table 5.1: Experimental results when G is C4.5RULES. Superscripts denote confidence levels: 1 is 99.5%, 2 is 99%, 3 is 97.5%, 4 is 95%, 5 is 90%, and 6 is below 90%.

Dataset	TWI-U	TWI-Y	C4.5R	RISE	Default
AD	78.9±4.3	66.3±6.6^{1}	70.6±5.7^{1}	77.3±4.9^{1}	21.3±2.6^{1}
BC	69.4±4.5	67.5±5.5^{4}	67.8±6.6^{5}	68.2±4.2^{3}	67.6±7.6^{6}
CE	83.5±1.7	84.2±2.9^{6}	84.8±2.5^{3}	83.1±2.0^{4}	57.4±3.8^{1}
DI	70.8±3.0	73.0±3.1^{2}	74.3±3.0^{1}	70.4±3.0^{4}	66.0±2.3^{1}
EC	68.1±4.9	67.3±8.2^{6}	65.9±7.6^{6}	67.4±4.9^{5}	67.8±6.6^{6}
GL	70.2±6.5	64.1±10.1^{2}	64.9±9.2^{1}	69.7±6.1^{6}	31.7±5.5^{1}
HD	79.8±3.7	76.6±3.1^{1}	77.8±4.3^{4}	79.6±3.9^{6}	55.0±3.4^{1}
HE	78.1±5.4	78.8±3.7^{6}	78.6±5.3^{6}	76.9±5.3^{3}	78.1±3.1^{6}
HO	83.8±3.5	81.2±6.3^{5}	81.2±4.4^{2}	81.9±3.2^{1}	63.6±3.9^{1}
IR	93.0±2.7	93.6±2.7^{6}	93.2±2.5^{6}	92.9±2.8^{6}	26.5±5.2^{1}
LA	83.7±9.9	81.6±10.7^{6}	86.3±8.6^{6}	89.2±10.6^{3}	65.0±9.5^{1}
LC	48.6±15.1	40.5±14.0^{4}	40.5±14.0^{4}	50.5±15.2^{5}	26.8±12.3^{1}
LD	64.0±5.6	65.8±4.6^{6}	64.4±3.9^{6}	63.4±5.4^{4}	58.1±3.4^{1}
LI	69.5±4.0	69.1±3.8^{6}	69.0±3.8^{6}	67.9±3.6^{3}	9.9±3.0^{1}
LY	78.4±6.2	76.9±3.9^{6}	75.6±4.9^{4}	80.2±6.8^{1}	57.3±5.4^{1}
PO	67.8±6.0	71.0±5.2^{2}	68.2±6.9^{6}	62.3±9.1^{1}	71.2±5.2^{2}
PR	87.7±5.8	80.6±10.1^{1}	80.4±10.0^{1}	87.7±5.5^{6}	43.1±4.2^{1}
PT	41.4±4.6	35.4±5.4^{1}	37.5±5.7^{1}	39.8±5.3^{1}	24.6±3.2^{1}
SF	71.9±3.1	71.2±4.1^{6}	71.1±4.1^{6}	70.8±2.7^{2}	25.2±4.4^{1}
SN	74.7±12.1	64.3±9.4^{1}	65.4±7.1^{1}	76.0±11.4^{3}	50.8±7.6^{1}
SO	83.1±6.8	75.1±5.8^{1}	78.9±5.9^{1}	84.8±6.5^{1}	9.1±2.1^{1}
VO	95.9±1.6	95.7±1.7^{6}	95.8±1.3^{6}	95.5±1.5^{5}	60.5±3.1^{1}
WI	96.3±2.3	91.3±5.6^{1}	91.8±5.6^{1}	96.9±1.8^{5}	36.4±9.9^{1}
ZO	93.5±3.8	90.0±5.2^{1}	89.6±4.7^{1}	93.2±3.7^{6}	39.4±6.4^{1}

Table 5.2: Experimental results when G is CN2. Superscripts denote confidence levels: 1 is 99.5%, 2 is 99%, 3 is 97.5%, 4 is 95%, 5 is 90%, and 6 is below 90%.

Dataset	TWI-U	TWI-Y	CN2	RISE	Default
AD	77.3±4.4	63.9±7.4^{1}	71.0±5.1^{1}	77.3±4.9^{6}	21.3±2.6^{1}
BC	68.4±6.3	67.9±6.4^{6}	67.9±7.1^{6}	68.2±4.2^{6}	67.6±7.6^{6}
CE	84.4±2.2	82.3±2.2^{1}	82.0±2.2^{1}	83.1±2.0^{1}	57.4±3.8^{1}
DI	72.3±2.7	73.7±2.9^{2}	73.8±2.7^{1}	70.4±3.0^{1}	66.0±2.3^{1}
EC	68.0±5.3	66.0±5.9^{3}	68.2±7.2^{6}	67.4±4.9^{6}	67.8±6.6^{6}
GL	69.9±6.8	58.9±6.3^{1}	63.8±5.5^{1}	69.7±6.1^{6}	31.7±5.5^{1}
HD	81.2±4.3	79.5±2.9^{3}	79.7±2.9^{4}	79.6±3.9^{2}	55.0±3.4^{1}
HE	79.6±5.3	80.5±4.5^{6}	80.3±4.2^{6}	76.9±5.3^{1}	78.1±3.1^{5}
HO	83.1±3.6	83.0±3.6^{6}	82.5±4.2^{6}	81.9±3.2^{1}	63.6±3.9^{1}
IR	93.4±2.7	93.2±2.8^{6}	93.3±3.6^{6}	92.9±2.8^{4}	26.5±5.2^{1}
LA	87.4±10.6	82.9±8.9^{3}	82.1±6.9^{4}	89.2±10.6^{6}	65.0±9.5^{1}
LC	42.3±14.5	39.1±14.8^{6}	38.6±13.5^{6}	50.5±15.2^{1}	26.8±12.3^{1}
LD	65.9±5.2	66.4±3.8^{6}	65.0±3.8^{6}	63.4±5.4^{1}	58.1±3.4^{1}
LI	69.5±4.0	68.9±4.5^{6}	69.5±3.8^{6}	67.9±3.6^{1}	9.9±3.0^{1}
LY	80.9±6.3	79.4±5.0^{5}	78.8±4.9^{4}	80.2±6.8^{6}	57.3±5.4^{1}
PO	68.3±4.9	65.0±6.9^{3}	60.8±8.2^{1}	62.7±9.2^{1}	71.2±5.2^{1}
PR	85.6±6.2	78.1±9.1^{1}	75.9±8.8^{1}	87.7±5.5^{5}	43.1±4.2^{1}
PT	41.4±5.2	38.9±5.1^{2}	39.8±5.2^{4}	39.8±5.3^{1}	24.6±3.2^{1}
SF	71.4±2.6	69.6±3.6^{2}	70.4±3.0^{5}	70.8±2.6^{6}	25.2±4.4^{1}
SN	73.8±10.6	63.2±9.0^{1}	66.2±7.5^{1}	76.0±11.4^{3}	50.8±7.6^{1}
SO	83.5±6.7	77.8±6.7^{1}	77.4±7.2^{1}	84.8±6.5^{1}	9.1±2.1^{1}
VO	95.0±1.8	95.5±1.5^{4}	95.8±1.6^{1}	95.5±1.5^{4}	60.5±3.1^{1}
WI	95.3±2.6	91.0±4.5^{1}	90.8±4.7^{1}	96.9±1.8^{1}	36.4±9.9^{1}
ZO	93.7±3.3	90.9±5.1^{1}	90.6±5.0^{1}	93.2±3.7^{5}	39.4±6.4^{1}

Table 5.3. Summary of TWI-U's results vs. other algorithms.

Measure	G = C4.5R			G = CN2		
	TWI-Y	C4.5R	RISE	TWI-Y	CN2	RISE
No. wins	18-6	17-7	17-6	20-4	19-4	16-7
No. signif. wins	11-2	12-2	11-4	14-2	13-2	10-5
Sign test	99.0	97.0	98.0	99.9	99.8	95.0

The results are also shown in bar graph form in Figure 5.6, where each pair of bars compares TWI-U with one of the other systems. In the left-hand group, C4.5RULES is used as the G component; in the right-hand one, CN2. As before, within each pair the left-hand bar represents the number of TWI-U wins vs. the other system, and the right-hand bar represents the number of the other system's wins. The darker portion of each bar represents the number of significant wins at the 95% confidence level.

The utility of two-way induction is clearly shown by these results, but it is important to understand why this advantage is observed. Inspection of the S and G rules shows that the S rules are indeed substantially more specific than the G ones. G rules typically contain a small number of conditions (approx. 1 to 5), whereas S rules often contain conditions on half of all the attributes. Further, tracing the execution of TWI-U reveals that by far the majority of S wins occur when the test example is also covered by a G rule, but G rules are still correct on most of the examples they cover. S rules thus encapsulate small exception areas to the broad generalizations represented by G rules, as intended.

TWI-U performed better than TWI-Y, and this was to be expected. TWI-Y's naive method of combining S and G rules (merge all) has several disadvantages. The G rules were not designed to be applied in a best-match manner. Being very general, they easily win examples outside them over more appropriate S rules. The S rules, conversely, were designed for best-match classification, but in the large sections of the instance space covered by G rules, they can only win examples that they actually cover.

TWI-U's accuracy is bounded from below by the fraction of cases on which the S and G components agree and are correct, and from above by 100% minus the fraction of cases on which they agree and are both incorrect. Further, when there are more than two classes, S and G components may differ and both be wrong. Extracting the cases where TWI-U's decision makes a difference, we see that it makes the correct decision approximately two-thirds of the time, on average. This is well above chance, but still leaves substantial room for improvement, at least in theory.

The study above was mainly concerned with accuracy. Two other variables of interest are speed and comprehensibility of the results. Compared to running its components separately, the time overhead added by TWI is minimal. Because

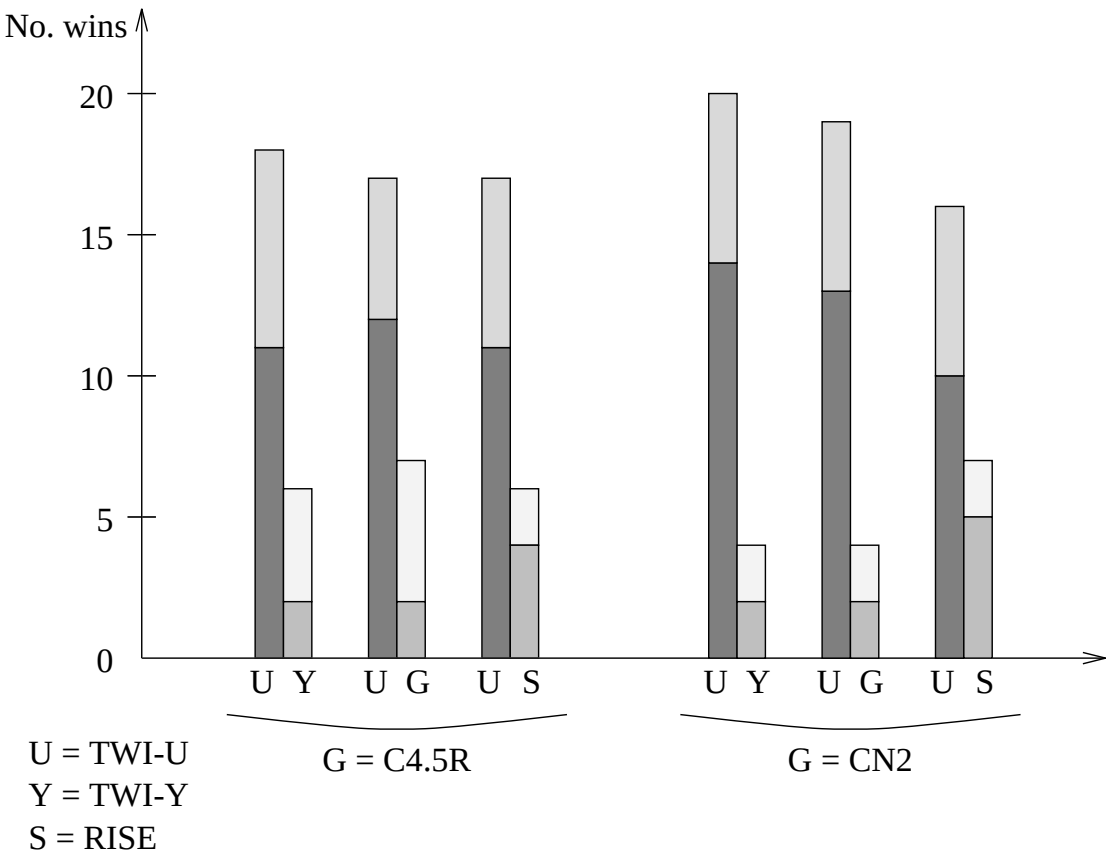

Figure 5.6: Comparison of TWI-U's accuracy with that of TWI-Y and the S and G components, when the G component is C4.5RULES (left) and CN2 (right).

the S rules are longer, they are harder for a human to understand; but since in TWI-U the distinction between S and G rules is maintained, the G rules can be seen as an approximate and accessible model of the domain, while the S rules represent second-order refinements and exceptions that should be taken into account to achieve higher accuracy.

Mitchell's (1982) version space approach is an early example of two-way induction. TWI is a heuristic algorithm, in contrast to Mitchell's exhaustive candidate elimination procedure, but has the advantage that its worst-case space and time complexities are respectively linear and low-order polynomial, instead of exponential. Also, unlike the candidate elimination algorithm, TWI is able to deal with disjunctive concepts, noise, and missing and continuous attributes.

5.4 Summary

Viewed as a rule induction system, RISE has a different bias from that of general-to-specific, "separate and conquer" learners like AQ, CN2 and C4.5RULES.

An initial study of this bias difference using artificial domains showed that RISE is better adapted to learning medium- to highly-specific concepts, but did not elucidate what conditions favor the more conventional rule inducers over it. Nevertheless, it is possible to take advantage of the complementarity of RISE's bias and that of these learners by designing an algorithm that combines the two. Even though this two-way induction algorithm still necessarily obeys Schaffer's (1994b) conservation law, it was observed to produce improvements in accuracy compared to the one-way approaches in a large sample of datasets from the UCI repository, showing it can be useful in practice.

Chapter 6
RISE as Instance-Based Learning

6.1 Overview

This chapter investigates empirically how RISE's behavior differs from that of other instance-based learning algorithms. The context-sensitive feature selection algorithm implicit in RISE is extracted from it, and compared with the classical methods of forward and backward selection. It is found to lead to higher accuracy in a large number of benchmark datasets. In experiments in artificial domains, its advantage is observed to increase with the degree of context dependency of feature relevance in the target concepts. RISE's feature selection algorithm is also much faster than forward and backward selection.

6.2 Context-Sensitive Feature Selection

If RISE has indeed a sensitivity to context that is absent from classical feature selection algorithms, as the simple example in Section 3.4 suggests, then its use should lead to significant improvements in accuracy when context effects are present. This advantage should increase with the degree of context dependency of feature relevance in the domain. On the other hand, when features are either globally relevant or globally irrelevant, RISE should have no advantage. Furthermore, if few examples are available or the data is noisy, context-free feature selection algorithms should be able to detect the globally irrelevant features more easily than RISE. This is due to the fact that they consider dropping a feature in all instances at once, instead of in one at a time, and so produce larger swings in accuracy, that can be more easily detected over statistical fluctuations when the examples are noisy and/or few. This chapter tests this hypothesized difference in bias between RISE and context-free feature selection algorithms, using both benchmark and artificial datasets.

Comparing RISE directly with FSS (forward sequential selection) and BSS (backward sequential selection) would unduly favor RISE, because RISE incorporates several mechanisms that go beyond feature selection: it can perform instance selection when there is duplication, expand points into intervals when dealing with

Table 6.1. The RC feature selection algorithm.

Input: ES is the training set.

Procedure RC (ES)

Let IS be ES.
Compute $Acc(IS)$.
Activate all instances in IS.
Repeat
  For each active instance I in IS,
    Find the nearest example E to I at nonzero distance, and of I's class.
    Let $I' = I$ with all features that differ in I and E removed.
    Let $IS' = IS$ with I replaced by I'.
    If $Acc(IS') \geq Acc(IS)$
      Then Replace IS by IS',
      Else Deactivate I.
Until all instances in IS are inactive.
Return IS.

numeric features, and take the interaction between induction steps into account when using a global stopping criterion. In order to make the comparison fair, it is thus necessary to remove all these aspects from RISE, ensuring that only the core of context-sensitive feature selection remains. The resulting simplified algorithm will be called RC (for Relevance in Context), and is summarized in pseudo-code in Figure 6.1. An instance I is an example with one or more features deleted, and corresponds to a rule in RISE. Recall that, in keeping with the idea of context-sensitive feature selection, different features can be present in different instances. The activation and deactivation of instances ensures that stopping is local (i.e., an instance stops being simplified as soon as there is no improvement in accuracy from simplifying it, without waiting to see if the simplification of other instances will reverse this).

A question that arises in RC is: when should two numeric values be considered different? If two real feature values are very similar but not identical, the fact that they differ should obviously not be construed as evidence that the feature is irrelevant. Thus it is necessary to decide where the critical point should be. The policy adopted was to compute the mean and standard deviation of each numeric feature from the sample in the training set, and attempt dropping the feature only when the values for the instance and the example differ by more than one standard deviation.

To understand this choice, suppose that the observed values of a feature fall into two or three clusters. Given any two values, if they differ by less than one

standard deviation they are most likely to be in the same cluster, and if they differ by more they are probably in different ones. Thus values in different clusters are judged to be significantly different, and only those. If there is a large number of clusters, the critical value should be less than one standard deviation. The choice made thus reflects a bias towards a small number of clusters; essentially, it assumes that the goal is either to distinguish between values above and below a certain threshold (the two-cluster case), or between a central range and values outside it (the three-cluster case). This is typically the case in many practical domains, like medical diagnosis (with variables like body temperature, blood pressure, levels of blood chemicals) and fault detection (voltage, stress, design dimensions with tolerances). In practice, an optimum value for this parameter can be determined by cross-validation, although this was not done in the studies described below.

The accuracy of an instance set $Acc(IS)$ is the fraction of the training examples that it correctly classifies using a leave-one-out methodology, as before. The pruning optimization used in RISE (see Section 3.5) was kept in RC.

6.3 Empirical Study: UCI Databases

To investigate empirically the hypothesis that RC's advantage increases with the context dependency of feature relevance, a measure of the latter is required. Unfortunately, in real-world domains the "true" degree of context dependency for a target concept is necessarily unknown. One way to circumvent this problem is to carry out studies in artificial domains, where the degree of context dependency can be predetermined by the experimenter, and this is done in Section 6.5. Another approach is to find an empirical measure that is thought to correlate positively with context dependency. One possibility is to find out how far RC strays from selecting the same features for all instances (i.e., from doing the same as FSS and BSS). More concretely, a possible measure is the average D for all pairs of instances of the number of features selected by RC for one but not the other:

$$D = \frac{2}{e(e-1)} \sum_{i=1}^{e} \sum_{j=1}^{i-1} \sum_{k=1}^{a} d_{ijk} \tag{6.1}$$

where e is the number of training examples, a is the number of features, and d_{ijk} is 1 if feature k was selected for instance i but not instance j or vice-versa, and 0 otherwise. This *feature difference* measure is necessarily imperfect, since the context dependency effects exhibited by RC may or may not be really present, but it is a legitimate one, in the sense that observing it can falsify the hypothesis that RC is more accurate relative to FSS and BSS when it detects greater context dependency. The core of the study that follows will thus be to correlate the feature difference D with the differential accuracy of RC and the context-free algorithms.

An empirical study was conducted using 24 datasets from the UCI repository (see Section 4.2). Twenty runs were carried out for each domain. In each, the training set was composed of two-thirds of the examples, chosen at random, and the remainder were used as test examples. The evaluation function used within FSS and BSS (see Tables 2.2 and 2.3) was the classifier's accuracy on the training set; it was measured using a leave-one-out methodology, as in RC. For each of the three algorithms (RC, FSS and BSS), the test-set accuracy obtained, running time and average number of features selected were recorded, and their averages for the 20 runs computed. Table 6.2 shows, for each domain, the feature difference D (Equation 6.1), the average accuracy and standard deviation for each algorithm, and the confidence that the difference between RC and each of the context-free algorithms is significant using a one-tailed paired t test. The domains are ordered by decreasing feature difference.

These results are presented in a more easily comprehended form in Figure 6.1, which shows the difference in accuracy between each of the context-free algorithms and RC as a function of feature difference. The straight lines shown are the result of linear regression on the empirical data points. The difference in accuracy between RC and BSS has a significant positive correlation with the feature difference (0.44), and similarly for FSS (0.36). We conclude that RC is indeed able to detect context dependency effects, and from the t test results in Table 6.2, that taking them into account in feature selection can produce significant improvements in accuracy.

Further analysis of the global results is shown in Table 6.3, and confirms the conclusion that RC is more accurate than FSS and BSS on this set of domains. The first line shows the number of domains in which RC achieved higher accuracy than the corresponding algorithm vs. the number in which the reverse occurred (e.g., RC was more accurate than BSS in 20 domains and less in 4). The second line shows the number of domains in which RC was more accurate than the other algorithm with a confidence level of 95% or higher, vs. the number in which the opposite occurred (i.e., in which the confidence is 5% or less). The results are very favorable to RC. The third line shows the confidence obtained applying a sign test to the values in line one. For example, RC's 20 wins vs. BSS have only a probability of occurrence of 1/1000. This results in very high confidence that RC is a more accurate algorithm than FSS and BSS on the population of domains from which the 24 used are drawn.

The gains obtained by using the context-sensitive algorithm, although consistent, are typically moderate (around 2% on average vs. BSS, and 3% vs. FSS). This is consistent with Holte's (1993) observation that, for some datasets in the UCI repository, accuracies within a small range of the best recorded values can be obtained using only the single most relevant feature. If RC, BSS and FSS all incorporate the "best" features, then their accuracies should not be expected to differ by more than this amount.

The number of features that each algorithm selects on average is also an indication of how the algorithms' behavior differs. It is reported in Table 6.4.

Table 6.2: Percentage accuracies of RC, FSS and BSS, and confidence levels for the difference between RC and FSS/BSS.

Dataset	Feature diff.	RC	FSS	Conf.	BSS	Conf.
LC	14.2±11.1	47.7±11.0	42.3±10.3	95.0	44.5±17.2	90.0
PR	8.6±14.3	89.1±6.0	84.4±8.9	95.0	84.9±5.6	99.0
HO	8.1±2.8	80.6±4.0	75.3±6.1	99.5	78.2±3.4	99.0
AD	6.8±4.1	77.0±4.9	71.2±5.7	99.5	75.4±4.4	97.5
VO	5.6±3.5	95.7±1.7	89.5±13.1	97.5	94.7±1.6	97.5
LA	5.0±2.0	91.1±6.9	87.4±6.7	95.0	85.8±9.2	99.0
PT	4.9±2.5	40.2±5.9	30.0±6.0	99.5	33.3±5.0	99.5
SO	4.4±2.9	100.0±0.0	94.4±8.8	99.0	95.0±6.3	99.5
HE	4.1±2.2	77.1±4.8	80.5±5.2	5.0	75.7±3.8	90.0
LY	3.8±1.8	81.2±5.6	76.5±5.2	99.5	79.1±6.0	95.0
ZO	3.1±1.9	93.2±3.8	91.0±5.1	95.0	90.3±5.6	99.5
SF	2.4±1.8	70.6±3.6	68.2±3.0	99.0	68.9±3.6	95.0
LI	2.1±1.3	61.4±6.2	47.1±13.3	99.5	54.7±7.8	99.5
CE	2.0±1.2	83.7±1.9	80.9±2.3	99.5	81.2±2.5	99.5
BC	1.6±1.1	66.2±5.2	66.7±6.7	40.0	66.9±6.1	35.0
HD	1.5±1.1	76.8±3.5	74.8±5.0	90.0	76.2±2.8	70.0
EC	0.8±1.0	60.2±6.1	59.4±5.2	60.0	60.3±5.6	50.0
PO	0.5±0.5	60.8±6.1	68.5±5.0	0.5	68.0±6.9	0.5
SN	0.5±0.3	81.4±9.1	73.5±11.2	99.5	80.5±8.7	85.0
LD	0.3±0.6	60.2±3.9	58.4±5.1	85.0	60.0±4.8	55.0
DI	0.2±0.5	70.5±2.5	69.6±2.9	80.0	69.2±3.3	97.5
GL	0.0±0.2	69.2±5.0	70.8±8.1	25.0	71.3±7.4	15.0
IR	0.0±0.0	94.4±2.4	92.6±2.3	99.5	92.9±2.9	97.5
WI	0.0±0.1	95.1±2.6	94.1±2.8	90.0	94.5±2.1	75.0

Table 6.3. Summary of accuracy results.

Measure	FSS	BSS
No. wins	20-4	20-4
No. signif. wins	15-2	14-1
Sign test	99.9	99.9

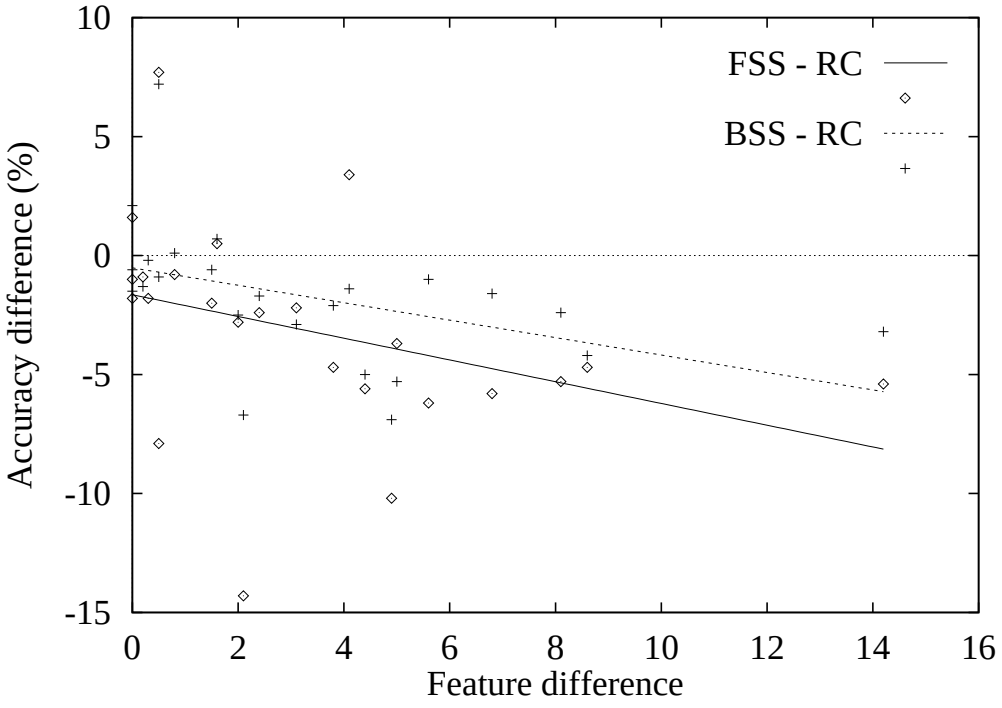

Figure 6.1. Empirical accuracy as a function of context dependency.

Since RC does not select the same set of features for all instances, its feature usage in each trial is defined as the average for all instances of the number of features used in each instance; for example, if a feature is used in only one of the e instances, it counts as only $1/e$ features. This is then averaged across all 20 trials. The average for all trials of the standard deviation *within each trial* of the number of features selected is also reported. As might be expected, it correlates positively with the feature difference, since a high value implies large variation in the features selected, even though the reverse is not true because two instances may have different features but the same number of features.

BSS always selects more features than FSS; this is not surprising, given their respective search strategies. RC almost always selects the most features,[1] but this observation can be misleading: direct inspection of the simplified instances output by RC shows that it typically drops most features from just a few of the instances, and it is these highly simplified ones that win most of the test examples; the majority of the instances retain most of the features, but have little impact on classification. Inspection also reveals that the most highly simplified instances differ in the features they retain. This, together with RC's higher accuracies, is

[1]The congressional voting records domain (VO) is an exception: here RC selects fewer features per instance than either FSS or BSS, and is also significantly more accurate than either. This suggests that strong context dependencies exist in this domain.

Table 6.4: Average number of features selected by the algorithms, and average feature difference of RC's instances.

Dataset	No. feats.	RC	FSS	BSS	Feature diff.
AD	69	64.1±4.4	11.4	22.0	6.8±4.1
BC	9	7.7±1.1	2.3	4.8	1.6±1.1
CE	15	13.5±1.2	5.7	9.6	2.0±1.2
DI	8	7.9±0.3	1.9	6.6	0.2±0.5
EC	7	6.6±0.8	1.5	4.4	0.8±1.0
GL	9	9.0±0.1	4.7	5.6	0.0±0.2
HD	13	12.1±0.9	4.6	9.3	1.5±1.1
HE	19	16.4±2.0	3.5	11.0	4.1±2.2
HO	22	14.8±3.4	5.7	15.6	8.1±2.8
IR	4	4.0±0.0	2.2	2.6	0.0±0.0
LA	16	11.3±2.0	3.0	6.8	5.0±2.0
LC	56	45.9±11.2	3.2	9.9	14.2±11.1
LD	6	5.9±0.5	2.1	4.2	0.3±0.6
LI	7	5.5±1.2	5.5	6.3	2.1±1.3
LY	18	15.2±1.9	5.2	11.3	3.8±1.8
PO	8	7.7±0.5	2.0	3.5	0.5±0.5
PR	57	52.1±11.7	5.4	16.1	8.6±14.3
PT	17	13.5±2.7	7.5	12.0	4.9±2.5
SF	12	10.1±1.9	4.4	5.7	2.4±1.8
SN	60	59.7±0.3	5.8	37.2	0.5±0.3
SO	35	25.2±1.5	2.0	3.9	4.4±2.9
VO	16	5.9±3.5	7.1	9.4	5.6±3.5
WI	13	13.0±0.1	4.3	8.4	0.0±0.1
ZO	16	13.2±1.8	5.8	6.4	3.1±1.9

further evidence that it is indeed detecting context sensitivity effects. However, if the goal is to reduce the feature set size as much as possible, even at some cost in accuracy, RC is clearly not the algorithm of choice: not only does it retain a higher number of features on average, but it only allows the removal of features that do not appear in any final instance.

The post-operative patient data domain (PO) is an example of a situation where FSS's and BSS's bias is more appropriate than RC's. In this domain, most features appear to be globally irrelevant; simply assigning all test examples to the most frequent class, ignoring all feature information, produces higher accuracy than all three algorithms (and also than decision-tree and rule learners; see Table 4.2). FSS and BSS correctly discard most of the features. However, because the dataset is small (90 examples) and noisy (as evinced by the fact that it contains inconsistent examples) RC has difficulty detecting the global irrelevance of features, and retains most of them for most instances.

6.4 Time Complexity

Another variable of interest is the running time of the algorithms. RC's worst-case running time is that of RISE with local stopping: $O(e^2a^2)$, where e is the number of examples and a is the number of features. The asymptotic time complexity of FSS and BSS can be computed as follows. The basic step of FSS/BSS consists of adding/deleting a single feature and checking the results. Since this involves comparing all instances with all examples along $O(a)$ features, the cost of each such step is $O(e^2a)$. This step is repeated for all currently excluded/included features and the best one is selected, which means that an $O(e^2a)$ step is repeated $O(a)$ times, resulting in a cost of $O(e^2a^2)$. Since in the worst case all features will be added/dropped, this cycle can be performed $O(a)$ times, resulting in a total cost of $O(e^2a^3)$.

However, this direct implementation of FSS and BSS is inefficient, because it unnecessarily repeats the computation of all distances along all features every time a feature is tentatively added or removed. A more efficient version will cache, for each example, the distances $\Delta(I, E)$ (Equation 3.1) of all instances to the example, and then, when considering adding or dropping a feature, add or subtract to each $\Delta(I, E)$ the distance component $\delta^2(i_j, e_j)$ along that feature. Once the example's predicted class is found and compared with the correct one, the original distance vector is reinstated, and the process repeats with the next feature. This implementation reduces the worst-case time complexity of FSS and BSS to $O(e^2a^2)$ (the same as RC's) and was the one used in the studies described here. Note that it does not require $O(e^2)$ memory instead of $O(e)$, because only the distances for one example at a time are cached. This implies bringing the cycle that classifies each example outside the cycle that tries each feature (i.e., "For each example, add/delete each feature and classify the example," instead of "For each feature,

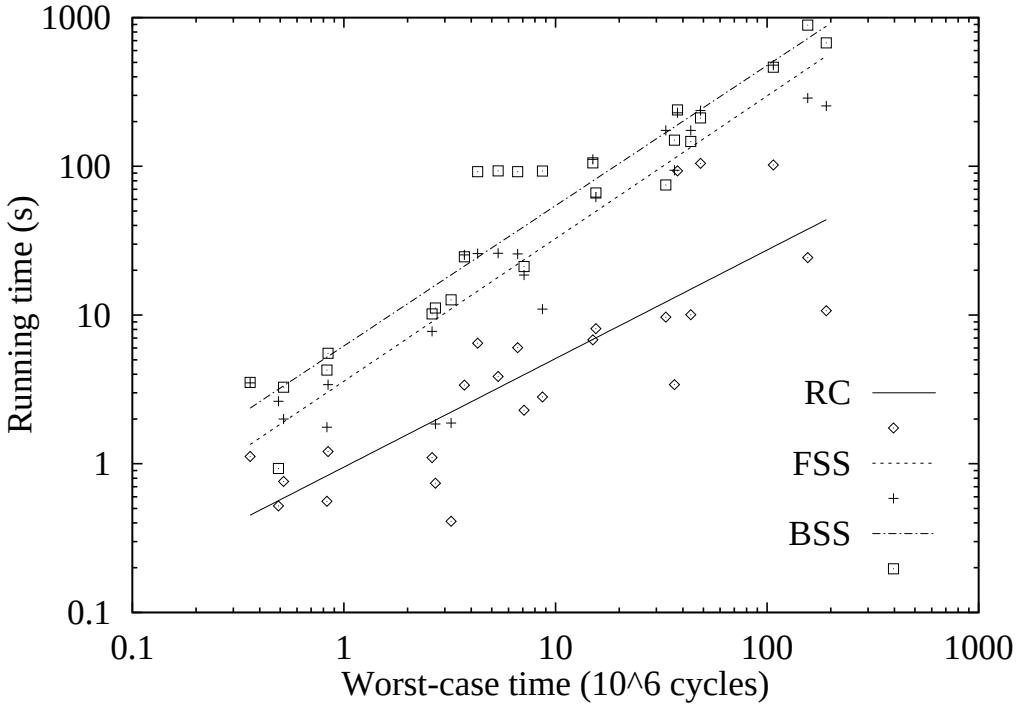

Figure 6.2. Running time in relation to e^2a^2.

add/delete the feature and classify each example"). Such a process, opening the "black box" of the classification algorithm and bringing the feature selection algorithm inside it, may not be possible for all instance-based algorithms. Also, it is incompatible with the pruning optimization used in RC (Section 3.5), and the latter could therefore not be applied in FSS and BSS.

Empirical running times are shown in Table 6.5. A Sun 670 workstation was used for all runs. Figure 6.2 shows these values plotted on a log-log scale against e^2a^2, the worst-case asymptotic growth rate for all algorithms. The straight lines shown are the result of linear regression on the log-log points. RC is always faster than FSS and BSS, sometimes by large factors (see, for example, the audiology domain, AD). This can be partly attributed to variations in the number of features that each algorithm actually adds/drops: since RC typically drops fewer features than BSS, and fewer than FSS adds, it finishes in fewer cycles. Other reasons for the observed difference in times are that the context-free algorithms have a higher multiplicative coefficient for e^2a^2, and the presence of additional lower-order terms in these. However, the most important factor is the optimization in the distance computation that was used in RC, but is not possible in the efficient versions of FSS and BSS.

Considering these effects, and extrapolating from the average slopes of the log-log plots of the algorithms' running times, we are led to hypothesize that

Table 6.5: Average running time of algorithms (in minutes and seconds), and ratio of running times of FSS and BSS to running time of RC.

Dataset	RC	FSS	BSS	FSS/RC	BSS/RC
AD	0:10.68	4:14.60	11:15.61	23.8	63.3
BC	0:06.03	0:25.78	1:31.95	4.3	15.2
CE	1:42.25	7:58.87	7:44.39	4.7	4.5
DI	1:33.12	3:48.80	3:59.52	2.5	2.6
EC	0:01.21	0:03.40	0:05.53	2.8	4.6
GL	0:03.38	0:25.39	0:24.63	7.5	7.3
HD	0:08.10	1:02.13	1:06.16	7.7	8.2
HE	0:02.81	0:10.97	1:32.94	3.9	33.1
HO	0:10.06	2:54.30	2:26.87	17.3	14.6
IR	0:01.12	0:03.50	0:03.52	3.1	3.1
LA	0:00.56	0:01.76	0:04.26	3.1	7.6
LC	0:00.41	0:01.88	0:12.64	4.6	30.8
LD	0:06.47	0:25.89	1:32.14	4.0	14.2
LI	0:00.52	0:02.63	0:00.93	5.1	1.8
LY	0:02.29	0:18.56	0:21.17	8.1	9.2
PO	0:00.76	0:02.00	0:03.27	2.6	4.3
PR	0:03.40	1:34.18	2:29.97	27.7	44.1
PT	0:09.68	2:54.47	1:14.92	18.0	7.7
SF	0:06.81	1:51.47	1:45.35	16.4	15.5
SN	0:24.32	4:47.99	14:49.84	11.8	36.6
SO	0:00.74	0:01.85	0:11.16	2.5	15.1
VO	1:44.70	3:56.94	3:31.63	2.3	2.0
WI	0:03.86	0:26.00	1:33.19	6.7	24.1
ZO	0:01.10	0:07.77	0:10.17	7.1	9.2

RC will still be a viable feature selection algorithm in domains where FSS's and BSS's time cost would exclude them from consideration. In the form used here, none of the algorithms are suitable for very large databases, since they are all necessarily quadratic in e, even in the average case; however, more efficient versions of FSS/BSS-style algorithms exist (Kittler, 1986; Aha & Bankert, 1994), and similar modifications of RC can be envisioned.

6.5 Empirical Study: Artificial Domains

The question arises of whether the conclusions formulated in the previous section are generally valid, or the favorable results obtained for RC are specific to the domains used in the study reported in the previous section. In other words, RC's observed benefits might apply only when the biases represented in the UCI repository are verified, independently of the more general hypothesis that they are due to RC's context sensitivity. Another question is whether the feature difference estimate used effectively corresponds to the context dependency we seek to measure, and thus whether the results obtained are meaningful. These two problems were investigated by carrying out experiments in artificial domains. The hypotheses to be tested are that RC is more accurate than FSS and BSS over a broad range of domains ("broad" in the sense that they have no common bias save their degree of context dependency), and that the difference in accuracy increases with the context dependency of feature relevance. In artificial domains, the target concept description is known *a priori*, and, if it is composed of a set of prototypes, the measure of feature difference defined in the previous section applied to that set of prototypes constitutes a suitable measure of context dependency. The empirical study thus proceeded by repeatedly selecting a value of D (Equation 6.1), generating a large number of domains at random characterized by that value, and observing the resulting accuracies of the three algorithms for that sample of domains.

Two-class problems were considered, with 100 examples in each dataset, described by 32 features. In each domain, each feature was chosen to be numeric or Boolean with equal probability (i.e., the number of numeric features is a binomial variable with expected value $a/2$ and variance $a/4$). Class 1 is defined by ten clusters, and class 0 is the complement of class 1. Each prototype or cluster is defined by a conjunction of conditions on the relevant features. The required value for a Boolean feature is chosen at random, with 0 and 1 being equally probable. Each numeric feature i must fall within a given range $[a_i, b_i]$, with a_i being the smaller of two values chosen from the interval $[-1, 1]$ according to a uniform distribution, and b_i the larger one. A cluster is thus a hyperrectangle in the relevant numeric subspace, and a conjunction of literals in the Boolean one.

The choice of relevant features for each prototype is made at random, but in a way that guarantees that the desired value of D for the set of prototypes is

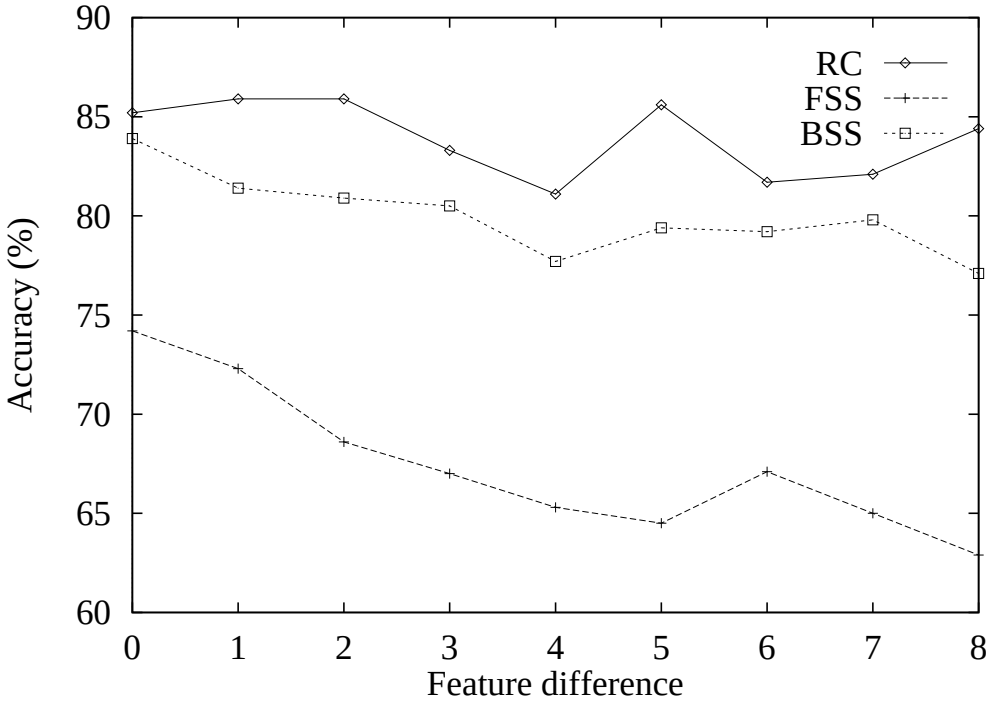

Figure 6.3. Accuracy as a function of context dependency.

maintained on average. The details of the procedure that does this are described in Appendix B. The feature difference D was varied from 0 to 8, the latter being the maximum value that can be produced given the number of features and prototypes used. Twenty domains were generated for each value of D, and two-thirds of the examples used as the training set. The average accuracy of RC, FSS and BSS on the remaining examples is shown graphically as a function of D in Figure 6.3.

All differences in accuracy between RC and FSS are significant at the 95% confidence level, as are those between RC and BSS for $D = 1$, 2, 4, 5, and 8. This confirms the hypothesis that RC is more accurate than FSS and BSS over a broad range of domains. However, BSS's performance is sometimes quite close to RC's. The smallest difference occurs when $D = 0$, as might be expected, since this situation exactly fits BSS's bias. More generally, due to the small number of training examples used (100), BSS may benefit from its ability to produce larger, more easily detected swings in accuracy when attempting to delete features, as previously hypothesized. Increasing the training set size should increase the distance between RC and BSS, since RC will then have enough data to detect the finer local dependencies that BSS by definition cannot. FSS's and BSS's time

limitations have precluded repeating the experiments with a significantly larger number of examples to investigate this point.[2]

The variation of the algorithms' accuracy with D is also of interest. All accuracies are negatively correlated with D, but the absolute value of the correlation is much smaller for RC (0.49) than for FSS and BSS (0.89 and 0.82, respectively). The downward slope of the regression line for RC's accuracy as a function of D (-0.35) is also much smaller than that for FSS (-1.21) and BSS (-0.61). We thus conclude that RC's higher performance is indeed at least partly due do its context sensitivity, and, pending further evidence, that using RC or RISE instead of conventional IBL will be justified whenever feature relevance is significantly context-dependent.

An additional hypothesis to explain RISE's advantage vs. "pure" IBL is that, because of its ability to simplify frontiers in the example space through abstraction, it can detect simple frontiers more easily when they exist. Another factor is the following. A number of IBL algorithms (e.g., PEBLS) discretize numeric features and then apply a VDM-style metric to the resulting symbolic descriptions. This improves their ability to cope with irrelevant features, since the VDM for two values of an irrelevant feature will tend to be zero, while Euclidean distance may be arbitrarily large. However, because discretization discards the ordering information present in numeric attributes, it also loses one of IBL's main advantages: its ability to form non-axis-parallel frontiers. Ting (1994) found that the discretization+VDM approach improved accuracy in a number of datasets, which can be attributed to the first effect having prevailed. In other datasets the opposite was observed. RISE overcomes this trade-off: it is still able to form non-axis-parallel frontiers through its use of Euclidean distance, and at the same time it is able to overcome irrelevant features by generalizing or entirely dropping them.

6.6 Related Work

Variations of FSS and BSS are described and evaluated in (Aha & Bankert, 1994). Beyond the pattern recognition approaches surveyed in (Devijver & Kittler, 1982) and (Kittler, 1986), many methods for feature selection have been proposed in the artificial intelligence literature in recent years (Almuallim & Dietterich, 1991; Kira & Rendell, 1992; Schlimmer, 1993; Vafaie & DeJong, 1993; Caruana & Freitag, 1994; John, Kohavi & Pfleger, 1994; Skalak, 1994). Cardie (1993) and Kibler and Aha (1987) use decision trees to select features for use in an instance-based learner. Although each path through the tree represents a context-dependent set of relevant features, this information is discarded, and only the unstructured

[2] With 100 examples, 20 runs with 9 values of D take approximately 10 hours of CPU time, of which less than 10 minutes is due to RC; 1000 examples would take on the order of $(1000/100)^2 \times$ 10 hours, or 40 days. On the other hand, reducing a to allow more examples without increasing time would further reduce the observable range of D.

set of all the features used in the tree is passed to the instance-based component. Another IBL feature selection method, also based on decision trees, is described in (Langley & Sage, 1994). In this case all paths through the tree contain the same set of features, and so the level of context sensitivity is similar to that of BSS.

A related field is that of feature weighting (Aha, 1989; Kelly & Davis, 1991; Salzberg, 1991; Creecy, Masand, Smith & Waltz, 1992; Mohri & Tanaka, 1994; Lowe, 1995). Feature selection can be seen as a special case of feature weighting where each weight is either 0 or 1, and thus weighting methods are potentially more powerful. However, because they have more degrees of freedom, they can also be harder to apply successfully, especially when there are few training examples. Feature weights can be supplied by the designer, as in Skalak's Broadway system (1992), or learned (see references above). Cain, Pazzani and Silverstein (1991) have an intermediate approach which combines instance-based and explanation-based learning, assigning higher weights to features that appear in the derivation of the example's class using a pre-existing domain theory.

Feature weighting methods also vary in what the weights can depend on, and thus in their degree of context sensitivity. In the representationally simplest schemes, there is one weight per feature, and they are therefore completely context-free (Kelly & Davis, 1991; Salzberg, 1991; Lee, 1994; Mohri & Tanaka, 1994; Lowe, 1995). More flexible approaches employ one weight per feature value (Nosofsky, Clark & Shin, 1989; Stanfill & Waltz, 1986), one weight per feature per class (Aha, 1989), or a combination of the two (Creecy, Masand, Smith & Waltz, 1992), and thus exhibit a moderate degree of context sensitivity. In the case of continuous features, it is also possible to take into account the relative values of the feature in the instance and the example being classified, resulting in directional weights (Ricci & Avesani, 1995). The most elaborate algorithms have in effect one weight per feature per instance, and are consequently fully context-sensitive; these weights can be assigned at classification time (Atkeson, Moore & Schaal, 1997; Hastie & Tibshirani, 1996) or at learning time (Aha & Goldstone, 1992). Seen as a 0-1 feature weighting algorithm, RC falls into this last category.

6.7 Summary

Viewed as an instance-based learner, RISE differs from conventional feature selection algorithms in that it has the ability to select different features for different instances (i.e., to select different features given different values of other features). Thus it can be expected to produce higher accuracies when feature relevance is significantly context-dependent. On the other hand, it can be at a disadvantage when no context effects are present, and the data is scarce and noisy. Both these trends were observed in an empirical study using benchmark datasets, with context-dependency being (at least apparently) present in a large majority of

the datasets, and RISE's feature selection algorithm reaping the corresponding rewards. It was also much faster than forward and backward selection. A study using artificial domains with controlled degrees of context dependency further confirmed that the difference in accuracy between RISE's feature selection and context-free methods increases as context dependency increases.

Chapter 7
Data Mining with RISE and CWS

7.1 Overview

Data mining seeks to extract useful knowledge from very large, and often very noisy, databases. RISE's quadratic running time makes it too slow for direct application to such problems. It can be made faster through the use of sampling techniques. Two of these are empirically studied in this chapter: windowing and partitioning. Their effect on running time and accuracy is measured in low- and high-noise situations. An alternative approach to achieving fast and robust induction is to design a linear-time algorithm incorporating RISE's "conquering without separating" strategy. The result, called CWS, is also described and evaluated in this chapter.

7.2 Data Mining: State of the Art

Very large datasets pose special problems for machine learning algorithms. A recent large-scale study found that most algorithms cannot handle such datasets in a reasonable time with a reasonable accuracy (Nakhaeizadeh, 1995; Michie, Spiegelhalter & Taylor, 1994). However, in many areas—including astronomy, molecular biology, earth sensing, finance, retail, marketing, health care, etc.—large databases are now the norm, and discovering patterns in them is a potentially very productive enterprise, in which interest is rapidly growing (Fayyad & Uthurusamy, 1995; Simoudis, Han & Fayyad, 1996). Designing learning algorithms appropriate for such problems has thus become an important research problem.

In these "data mining" applications, the main consideration is typically not to maximize accuracy, but to extract useful knowledge from a database. The learner's output should still represent the database's contents with reasonable fidelity, but it is also important that it be comprehensible to users without machine learning expertise. In this respect, RISE and other rule induction algorithms have the advantage that the "IF ... THEN ..." rules they produce are perhaps the most easily understood of all representations currently in use.

A major problem in data mining is that the data is often very noisy. Besides making the extraction of accurate rules more difficult, this can have a disastrous effect on the running time of rule learners. In C4.5RULES, noise can cause running time to become cubic in e, the number of examples (Cohen, 1995). When there are no numeric attributes, C4.5 has complexity $O(ea^2)$, where a is the number of attributes (Utgoff, 1989b), but its running time in noisy domains is dwarfed by that of the conversion-to-rules phase (Cohen, 1995). Outputting trees directly has the disadvantage that they are typically much larger and less comprehensible than the corresponding rule sets.

In algorithms that let the rule set grow to fit the data completely, and then simplify it by reduced error pruning (Brunk & Pazzani, 1991), the presence of noise causes running time to become $O(e^4 \log e)$ (Cohen, 1993). Fürnkranz and Widmer (1994) have proposed *incremental reduced error pruning (IREP)*, an algorithm that prunes each rule immediately after it is grown, instead of waiting until the whole rule set has been induced. Assuming the final rule set is of constant size, IREP reduces running time to $O(e \log^2 e)$, but its accuracy is often lower than C4.5RULES's (Cohen, 1995). Cohen introduced a number of modifications to IREP, and verified empirically that RIPPERk, the resulting algorithm, is competitive with C4.5RULES in accuracy, while retaining an average running time similar to IREP's (Cohen, 1995).

Catlett (Catlett, 1991) has done much work in making decision tree learners scale to large datasets. A preliminary empirical study of his peepholing technique shows that it greatly reduces C4.5's running time without significantly affecting its accuracy.[1] To the best of our knowledge, peepholing has not been evaluated on any large real-world datasets, and has not been applied to rule learners.

A number of algorithms achieve running time linear in e by forgoing the greedy search method used by the learners above, in favor of exhaustive or pruned near-exhaustive search (e.g., (Weiss, Galen & Tadepalli, 1987; Goodman, Higgins, Miller & Smyth, 1992; Segal & Etzioni, 1994)). However, this causes running time to become exponential in a, leading to a very high cost per example, and making application of those algorithms to large databases difficult. Holte's 1R algorithm (Holte, 1993) outputs a single tree node, and is linear in a and log-linear in e, with the logarithmic factor due to sorting of numeric values, but its accuracy is often much lower than C4.5's.

Ideally, we would like to have an algorithm capable of inducing accurate rules in time linear or near-linear in e, without becoming too expensive in other factors. One way of attempting to achieve this is to apply sampling techniques to an existing algorithm, thus reducing its running time, in the hope that this will not have too negative an effect on accuracy. The alternative is to design a new algorithm with speed explicitly in mind, but incorporating at least some of the

[1] Due to the small number of data points (3) reported for the single real-world domain used, it is difficult to determine the exact form of the resulting time growth (linear, log-linear, etc.).

useful characteristics of existing algorithms. Both approaches are explored in this chapter, with RISE as the starting point.

7.3 Speeding Up RISE

Windowing in RISE was described in Section 3.5. In the partitioning speedup approach (Chan & Stolfo, 1995b), the training data is divided into a number of disjoint subsets, and the learning algorithm is applied to each in turn. The results of each run are combined in some fashion, either at learning or at classification time. In RISE, partitioning is applied by pre-determining a maximum number of examples e_{max} to which the algorithm can be applied at once (100 by default). When this number is exceeded, the training set is randomly divided into $\lceil e/e_{max} \rceil$ approximately equal-sized partitions, where e is the total number of training examples. RISE is then run on each partition separately, but with an important difference relative to a direct application: the rules grown from the examples in partition p are not evaluated on the examples in that partition (see Table 3.1 and accompanying discussion), but on the examples in partition $p + 1$, modulo the number of partitions. This should help combat overfitting, and the resulting improvement in accuracy may partly offset the degradation potentially caused by using smaller training sets. It is not possible in systems like C4.5RULES and CN2, where there is no connection between a specific rule and a specific example.

Because the number of partitions grows linearly with the number of training examples, and RISE's quadratic factor is confined to the examples within each partition and thus cannot exceed a given maximum (e.g., 100^2 if $e_{max} = 100$), the algorithm with partitioning is guaranteed a linear worst-case running time. However, depending on e_{max}, the multiplicative constants can become quite large.

Three methods of combining the results of induction on the individual partitions have been implemented and empirically compared. In the first, all the rule sets produced are simply merged into one, which is output by the learning phase. In the second, the rule sets are kept separate until the performance phase, and each partition classifies the test instance independently. A winning class is then assigned to the example by voting among the partitions, with each partition's weight being the Laplace accuracy of the rule that won within it (Equation 2.2). In the third method, the rule sets are again kept separate, and Bayesian model averaging is employed, in a form similar to that of (Buntine, 1990) and (Ali & Pazzani, 1996). Appendix C describes the details of this procedure. The second method was found to achieve consistently better results, and was therefore adopted. The reasons for this heuristic approach outperforming the more principled Bayesian one are explored in (Domingos, 1997a). Many other combination schemes are possible (e.g., (Chan & Stolfo, 1995b)).

Windowing and partitioning were tested on seven of the UCI repository's largest databases (in increasing order of size: credit, Pima diabetes, annealing,

Table 7.1. Experimental results: running times (in minutes and seconds).

Database	RISE	Windowing	Partitioning		
			$e_{max}=100$	$e_{max}=200$	$e_{max}=500$
CE	4:31	3:21	1:37	1:11	4:38
DI	4:15	6:20	1:32	1:13	2:47
AN	4:26	2:44	1:43	2:33	2:17
CH	33:26	10:40	3:10	6:04	12:06
HY	105:23	14:46	5:08	10:42	24:06
SP	110:39	51:28	5:22	12:45	25:48
MU	70:07	10:07	5:55	7:26	14:32

chess endgames, hypothyroid, splice junctions, and mushroom; see Section 4.2). Of these, at least one (Pima diabetes) is thought to be quite noisy, and at least two (chess and mushroom) are known to be almost entirely noise-free. Partitioning was tested with $e_{max}=100$, 200, and 500. Ten runs were carried out for each database, in each run randomly dividing the data into two-thirds for training and one-third for testing. The averaged results are shown in Tables 7.1 (running times) and 7.2 (accuracies). They are also presented in bar graph form in Figures 7.1 and 7.2.

Both speedup methods are effective in reducing RISE's running time, generally without seriously affecting accuracy (chess and annealing with partitioning, and Pima diabetes with windowing, are the exceptions). Windowing often has practically no effect on accuracy. Thus, overall this method appears to be more useful in RISE than in decision tree induction (Catlett, 1991). This may be due to several factors, including RISE's lower sensitivity to the global proportions of different classes, and its higher resistance to the fragmentation problem, which enables it to correctly approximate class frontiers using fewer examples.

Partitioning's running time is (as might be expected) sensitive to the choice of e_{max}, but it appears to increase less than linearly with it. Linear growth would

Table 7.2. Experimental results: accuracies and standard deviations.

Database	RISE	Windowing	Partitioning		
			$e_{max}=100$	$e_{max}=200$	$e_{max}=500$
CE	82.6±1.5	83.6±1.5	86.4±1.9	86.4±1.5	82.6±1.6
DI	71.6±2.5	70.6±2.7	74.4±2.1	73.6±3.3	72.8±2.6
AN	97.5±0.9	98.0±1.0	93.6±1.6	96.1±1.6	96.5±1.1
CH	98.4±0.6	98.4±0.7	94.5±0.5	95.2±0.6	96.6±0.9
HY	97.9±0.2	97.5±0.5	97.0±0.3	97.5±0.3	97.9±0.4
SP	92.5±0.8	92.8±0.7	95.0±0.7	94.6±0.7	94.7±0.6
MU	100.0±0.0	100.0±0.0	98.9±0.1	99.5±0.3	99.8±0.1

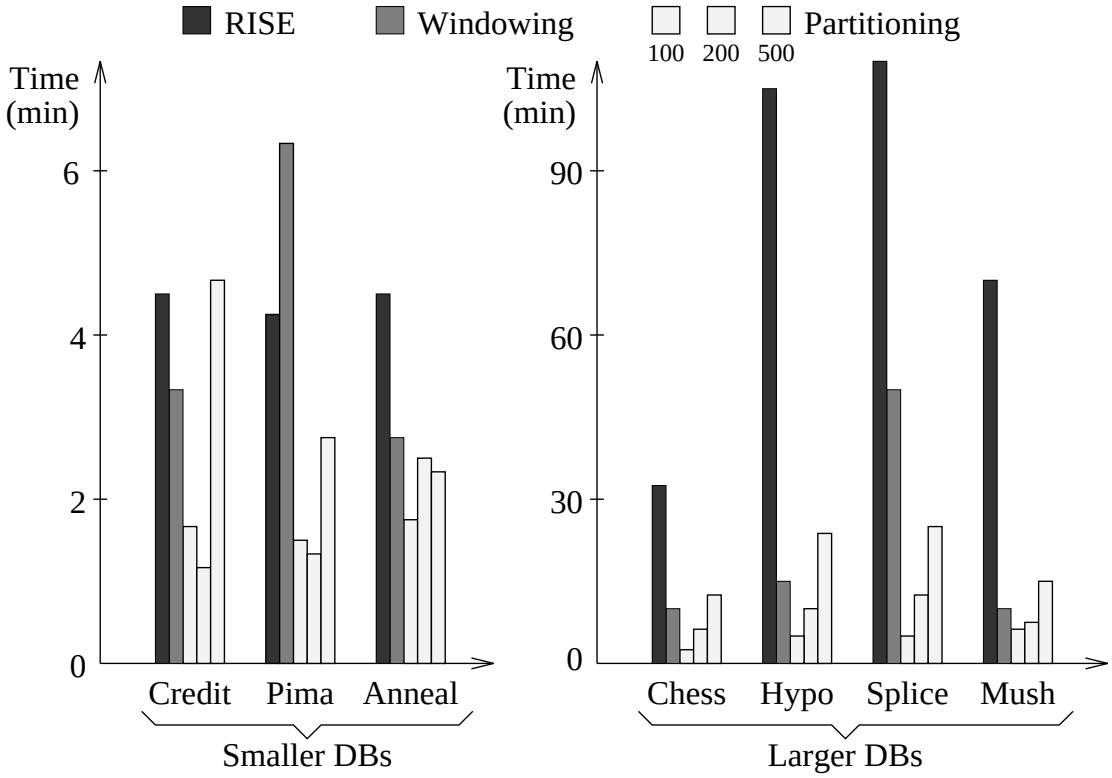

Figure 7.1: Comparison of running times: RISE, RISE with windowing and RISE with partitioning.

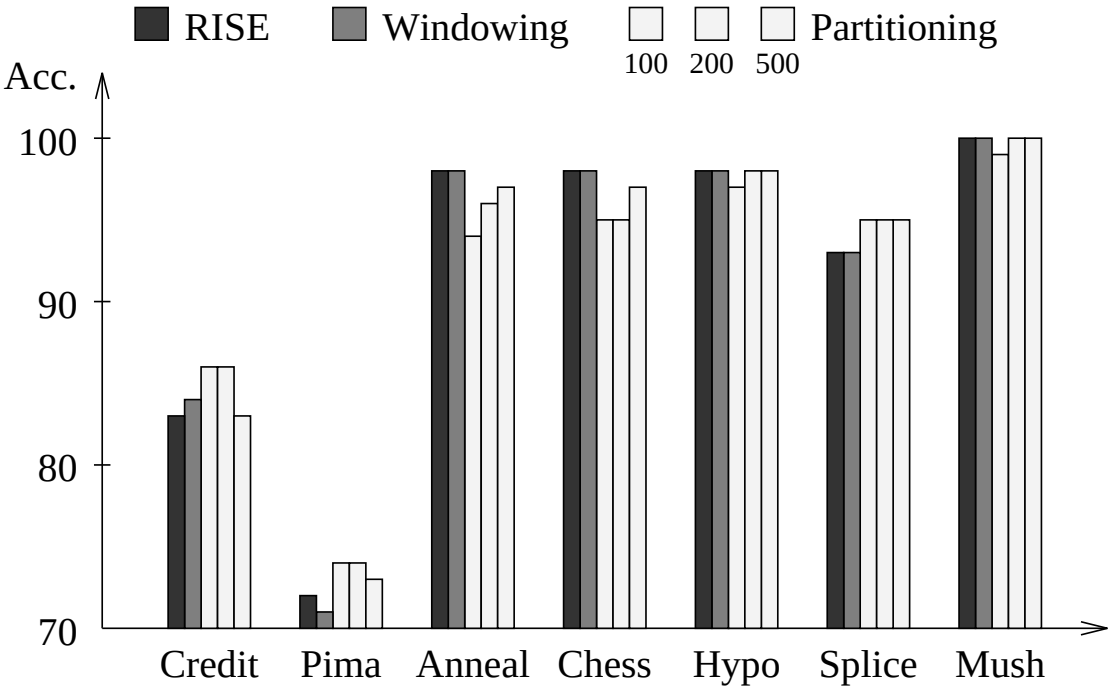

Figure 7.2: Comparison of accuracies: RISE, RISE with windowing and RISE with partitioning.

be expected, since, if p is the number of partitions and t is the total running time, $t = O(pe_{max}^2)$, and since $p \simeq e/e_{max}$, $t = O(ee_{max})$, i.e., for a given e, $t \propto e_{max}$. Examination of the rules produced shows that RISE tends to stop earlier within each partition when the partitions are larger, presumably because the additional information available warrants the induction of more specific rules, and while in general-to-specific systems this means that the algorithm will take longer to run because more antecedents will be added, in RISE the opposite is the case, since fewer antecedents will be deleted. This will tend to partly offset the increase in running time due to increasing partition size.

The effect of partitioning on accuracy is more variable than that of windowing. In some domains a trade-off between partition size and accuracy is observed; however, only in the chess domain does increasing e_{max} from 200 to 500 substantially increase accuracy. More interestingly, in the credit, diabetes and splice junctions domains the opposite trend is observed (i.e., partitioning increases accuracy, and smaller partitions more so than larger ones); this may be attributed to the reduction in overfitting derived from inducing and testing rules on different partitions, to the increase in accuracy that can result from combining multiple models (Wolpert, 1992; Breiman, 1996a; Breiman, 1996c), and possibly to other factors. On the splice junctions dataset, the success of applying partitioning to RISE using a simple combination scheme contrasts with the results obtained by Chan and Stolfo for general-to-specific learners (Chan & Stolfo, 1995a). In general, the best partition size should be determined by experimentation on the specific database RISE is being applied to, starting with smaller (and therefore faster) values.

To test the algorithms on a larger problem, and obtain a clearer view of the growth rate of their running times, experiments were conducted on NASA's space shuttle database (Catlett, 1991). This database contains 43500 training examples from one shuttle flight, and 14500 test examples from a different flight. Each example is described by nine numeric attributes obtained from sensor readings, and there are seven possible classes, corresponding to states of the shuttle's radiators. The goal is to predict these states with very high accuracy (99–99.9%), using rules that can be taught to a human operator.

Figure 7.3 shows the evolution of running time with the number of examples for RISE, RISE with partitioning (using $e_{max} = 100$), and RISE with windowing, on a log-log scale. Recall that on this type of scale the slope of a straight line corresponds to the exponent of the function being plotted. Canonical functions approximating the asymptotic curves for RISE and RISE with partitioning are also shown.[2] Windowing reduces running time, but its growth appears to remain roughly quadratic; partitioning reduces it to linear, as expected. On the full training database, RISE consumes over a week of CPU time, while partitioning takes less than an hour. The accuracy curves (not shown) are very similar for all systems,

[2]The constants a and b were chosen so as to make the respective curves fit conveniently in the graph.

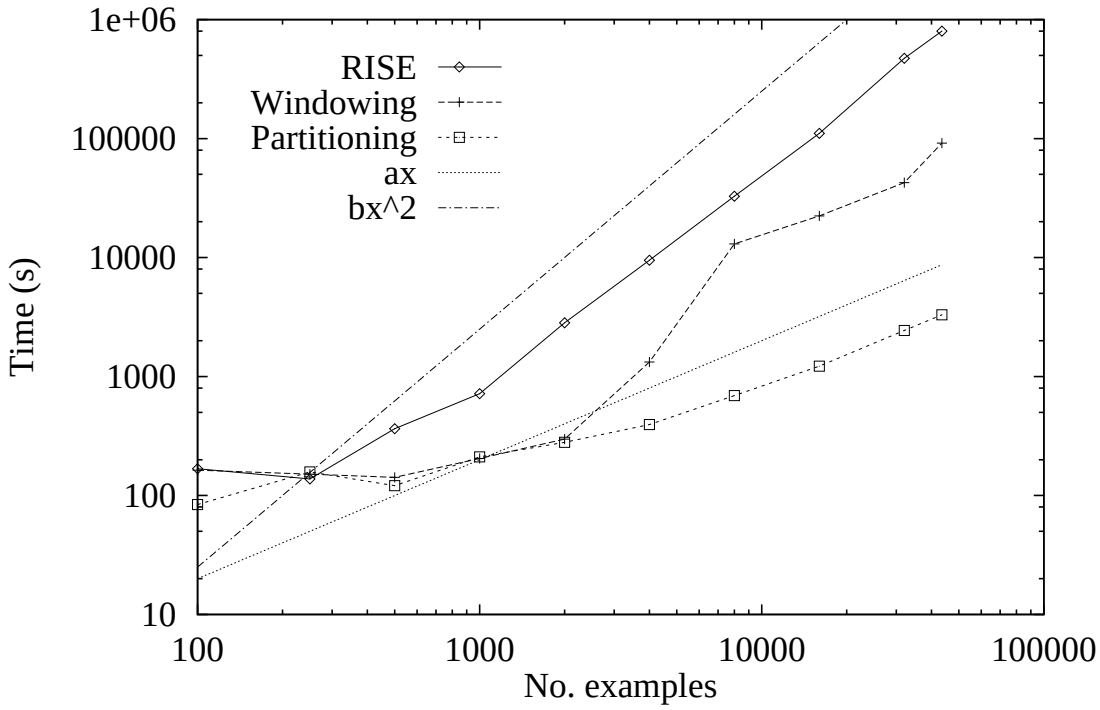

Figure 7.3. Learning times for the shuttle database.

converging rapidly to very high values (99% by $e = 1000$, etc.), with partitioning lagging slightly behind the other two (0.55% on average, with respect to RISE).

The shuttle data is known to be relatively noise-free. To investigate the effect of noise, the three algorithms (pure RISE, partitioning and windowing) were also applied after corrupting the training data with 20% class noise (i.e., each class had a 20% probability of being changed to a random class, including itself). The learning time curves obtained are shown in Figure 7.4, again on a log-log scale and with approximate asymptotes shown. The time performance of windowing degrades markedly, making it worse than the pure algorithm for all training set sizes greater than 500. In contrast, partitioning remains almost entirely unaffected. Noise reduces the accuracy of pure RISE and windowing by 3 – 8%, with the smaller differences occurring for larger training set sizes. (Recall that noise was added only to the training set.) The accuracy of partitioning is barely affected, making it consistently more accurate than pure RISE at this noise level.

An interesting observation is that noise substantially reduces RISE's running time, even though it remains much larger than that obtained with partitioning. This may be attributed to the difference between specific-to-general and general-to-specific systems already discussed: noise tends to make rule induction algorithms produce more specific rules, which take less time to induce in RISE and more in systems like C4.5RULES (which, in addition, may then prune back those rules,

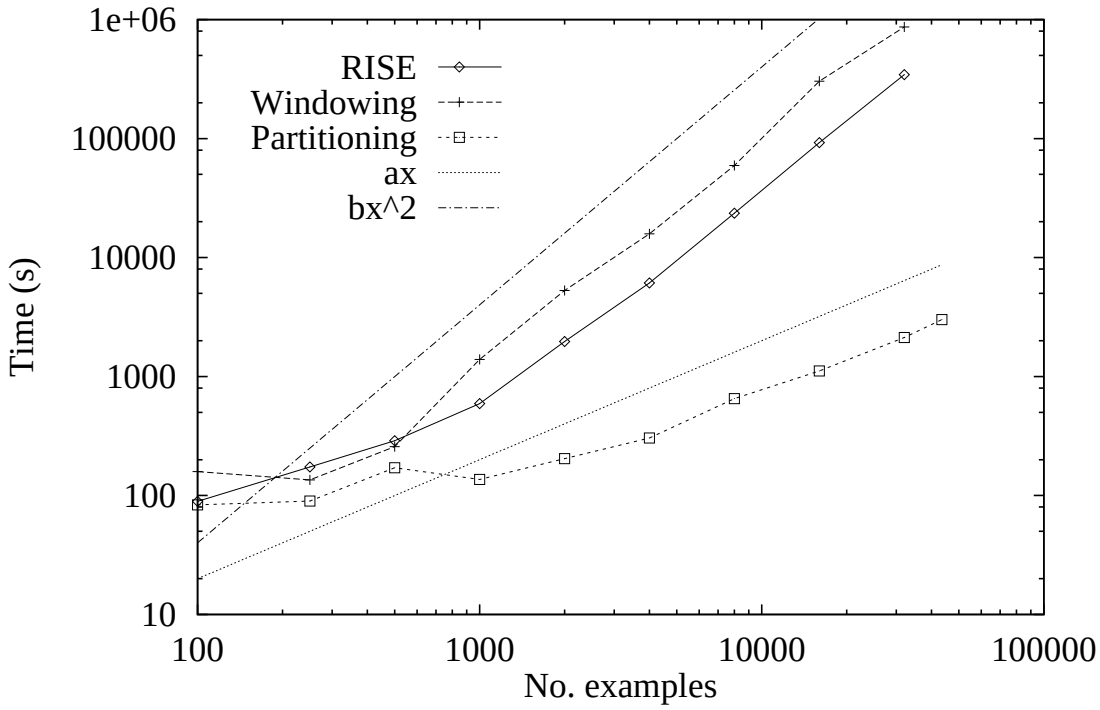

Figure 7.4. Learning times for the shuttle database with 20% noise.

further adding to their running time; in RISE pruning and initial induction are the same operation). This means that RISE may be more appropriate than general-to-specific systems for noisy databases.

A potential disadvantage of partitioning when compared to windowing is that it sometimes (but not always) tends to produce rule sets that are larger overall, even if the individual rule sets learned from each partition are comparatively small. However, from the point of view of output comprehensibility, this is not necessarily a serious problem, since understanding can still be gleaned by looking at one of the individual rule sets, or one at a time.

7.4 The CWS Algorithm

In the "separate and conquer" methodology employed by most rule learners, each rule is induced to its full length before going on to the next one, and each rule is evaluated by itself, without regard to the effect of other rules. Apart from its effect on accuracy, this approach is potentially inefficient: rules may be grown further than they need to be, only to be pruned back afterwards, when the whole rule set has already been induced. This will be particularly likely in noisy domains, and Cohen (1993) has shown that it is indeed the cause of the very high running times

he observed. In contrast, RISE interleaves the construction of all rules, evaluating each rule in the context of the whole rule set, and is thus able to achieve high accuracies without running into this problem. The main source of inefficiency in RISE is not overfitting and post-pruning, but the repeated computation of distances between rules and examples. Since the number of rules is initially the number of examples, and each rule is initially compared with all examples, a super-linear running time is inevitable.

However, this problem can be avoided if RISE's "conquering without separating" approach is applied in a general-to-specific setting, with interleaved rule induction and global evaluation as before, but with an initially empty rule set and operators that add antecedents to a rule, as in more conventional systems. With careful optimization of the type described in the last paragraph of Section 3.3, this holds the promise of leading to lower running times than either RISE or "separate-and-conquer" systems, while still achieving higher accuracies than the latter by reducing the fragmentation and small disjuncts problems.

CWS (for "Conquering Without Separating") is an algorithm that instantiates this idea. CWS's rules differ from RISE's in three aspects. First, they have a purely logical interpretation: examples not covered by any rule are assigned to the default class (the class with the most examples in the training set[3]). This avoids the distance computations that result in RISE's quadratic running time. Second, conditions on numeric attributes are inequalities of the form $a_i > v_{ij}$ or $a_i < v_{ij}$, as in CN2 and C4.5RULES, instead of intervals. Use of the latter, with the corresponding consideration of all pairs of endpoints at each induction step, would result in a running time quadratic in the number of values per attribute. Lastly, each rule in CWS is also associated with a vector of class probabilities computed from the examples it covers, the predicted class being the one with the highest probability. For class C_i, $P_r(C_i)$ is estimated by e_{ri}/e_r, where e_r is the total number of examples covered by rule r, and e_{ri} is the number of examples of the ith class among them. When a test example is covered by more than one rule, the class probability vectors of all the rules covering it are summed, and the class with the highest sum is chosen as the winner. This is similar to the approach followed in CN2 (Clark & Boswell, 1991), with the difference that probabilities are used instead of frequencies. In a preliminary study, the use of probabilities was found to lead to higher accuracies. This can be attributed to the frequency-based approach's tendency to assign excessive weight to rules with large coverage, making it difficult for rules representing smaller exception regions to ever have an effect.

This voting approach to conflict resolution allows an optimization similar to that used in RISE (see Section 3.3), which would not be possible in a general-to-specific induction setting with RISE's "choose one winner" approach. To see this, consider that, when a rule is generalized in RISE, it can become the new

[3] An alternative would be to take as default the class with the most training examples not covered by any rule. However, this would add complexity to the learning algorithm without any clear gain, because in CWS the default class chosen by either method tends to be the same.

Table 7.3. The CWS algorithm.

Input: ES is the training set.

Procedure CWS (ES)

Let $RS = \emptyset$.
Repeat
  Add one active rule R_{new} with empty body to RS.
  For each active rule R in RS,
    For each possible antecedent AV,
      Let $R_{AV} = R$ with AV conjoined to its body.
      Compute class probabilities and predicted class for R_{AV}.
      Let $RS_{AV} = RS$ with R replaced by R_{AV}.
      Let $\Delta_{AV} Acc(RS) = Acc(RS_{AV}) - Acc(RS)$.
    Pick AV' with maximum $\Delta_{AV} Acc(RS)$.
    If $\Delta_{AV'} Acc(RS) > 0$
      Then $R = R_{AV'}$,
      Else Deactivate R.
  If R_{new} has been deactivated
    Then Delete it.
Until all rules in RS are inactive.
Return RS.

closest rule to an example, but if it already is the closest rule, it cannot lose that position, because generalization can only reduce a rule's distance to an example. Thus, in order to evaluate the effect of a rule generalization on the rule set's accuracy, all that is needed is to check whether the rule wins any new examples, and what its effect on those is. If each example's distance to its nearest rule is memorized, this requires only $O(ea)$ time (where e is the number of examples, and a is the number of attributes), while matching all r rules would require $O(rea)$. Conversely, in general-to-specific induction, when a rule is specialized it cannot gain new examples, but it can lose some that it previously won. If a "choose one winner" conflict resolution policy is used, it then becomes necessary to find the new winning rule, and this requires matching all the existing r rules with the "lost" examples, leading to an $O(rea)$ operation. In contrast, when a voting approach is used, all that is needed is to subtract the rule's votes from the examples it no longer covers, and this can be done efficiently, as detailed below.

CWS is outlined in pseudo-code in Table 7.3. Initially the rule set is empty, and all examples are assigned to the majority class. In each cycle a new rule with empty body is tentatively added to the set, and each of the rules already there is specialized by one additional antecedent. Thus induction of the second rule starts

immediately after the first one is begun, etc., and induction of all rules proceeds in step. At the end of each cycle, if a rule has not been specialized, it is deactivated, meaning that no further specialization of it will be attempted. A rule with empty body predicts the default class, but this is irrelevant, because a rule only starts to take part in the classification of examples once it has at least one antecedent, and it will then predict the class that most training examples satisfying that antecedent belong to. A rule's predicted class may change as more antecedents are added to it. $Acc(RS)$ is the accuracy of the rule set RS on the training set, as before.

Let e be the number of examples, a the number of attributes, v the average number of values per attribute, c the number of classes, and r the number of rules produced. The basic step of the algorithm involves adding an antecedent to a rule and recomputing $Acc(RS_{AV})$. This requires matching all rules with all training examples, and for each example summing the class probabilities of the rules covering it, implying a time cost of $O[re(a+c)]$. Since there are $O(av)$ possible antecedents, the cost of the inner loop ("For each AV", see Table 7.3) is $O[avre(a+c)]$. However, this cost can be much reduced by avoiding the extensive redundancy present in the repeated computation of $Acc(RS_{AV})$. The key to this optimization is to avoid rematching all the rules that remain unchanged when attempting to specialize a given rule, and to match the unchanged antecedents of this rule with each example only once. Recomputing $Acc(RS_{AV})$ when a new antecedent AV is attempted now involves only checking whether each example already covered by the rule also satisfies that antecedent, at a cost of $O(e)$, and updating its class probabilities if it does, at a cost of $O(ec)$. The latter term dominates, and the cost of recomputing the accuracy is thus reduced to $O(ec)$, leading to a cost of $O(eavc)$ for the "For each AV" loop.

In more detail, the optimized procedure is as follows. Let $Cprobs(R)$ denote the vector of class probabilities for rule R, and $Cscores(E)$ denote the sum of the class probability vectors for all rules covering example E. $Cscores(E)$ is maintained for each example throughout. Let R be the rule whose specialization is going to be attempted. Before the "For each AV" loop begins, R is matched to all examples, those which satisfy it are selected, and, for each such example E, $Cprobs(R)$ is subtracted from $Cscores(E)$. $Cscores(E)$ now reflects the net effect of all other rules on the example. Each possible antecedent AV is now conjoined to the rule in turn, leading to a changed rule R_{AV}. New class probabilities for R_{AV} are computed by finding which examples E_{AV} among the previously-selected ones satisfy AV. These probabilities are now added to $Cscores(E_{AV})$ for the still-covered examples E_{AV}. Examples that were uncovered by the specialization already have the correct values of $Cscores(E)$, since the original rule's $Cprobs(R)$ were subtracted from them beforehand. All that remains is to find the new winning class for each example E. If the example was previously misclassified and is now correctly classified, there is a change of $+1/e$ in the accuracy of the rule set. If it was previously correctly classified and is now misclassified, the change is $-1/e$. Otherwise there is no change. Summing this for all the examples yields the global

change in accuracy. As successive antecedents are attempted, the best antecedent and maximum global change in accuracy are remembered. At the end of the loop the best antecedent is permanently added to the rule, if the corresponding change in accuracy is positive. This simply involves repeating the procedure above, this time with permanent effects. If no antecedent produces a positive change in accuracy, the rule's original class probabilities $Cprobs(R)$ are simply re-added to the $Cscores(E)$ of all the examples that it covers, leaving everything as before. This procedure is shown in pseudo-code in Table 7.4. Note that the optimized version produces exactly the same output as the non-optimized one; conceptually, the much simpler Table 7.3 is an exact description of the CWS algorithm.

The total asymptotic time complexity of the algorithm is obtained by multiplying $O(eavc)$ by the maximum number of times that the double outer loop ("Repeat ... For each R in RS ...") can run. Let s be the output size, measured as the total number of antecedents effectively added to the rule set. Then the double outer loop runs at most $O(s)$, since each computation within it (the "For each AV" loop) adds at most one antecedent. Thus the total asymptotic time complexity of CWS is $O(eavcs)$.

The factor s is also present in the complexity of other rule induction algorithms (CN2, IREP, RIPPERk, etc.), where it can typically grow to $O(ea)$. It can become a significant problem if the dataset is noisy. However, in CWS it cannot grow beyond $O(e)$, because each computation within the double outer loop ("Repeat ... For ...") either produces an improvement in accuracy or is the last one for that rule, and in a dataset with e examples at most e improvements in accuracy are possible. Ideally, s should be independent of e, and this is the assumption made in (Fürnkranz & Widmer, 1994) and (Cohen, 1995), and verified below for CWS.

CWS incorporates three methods for handling numeric values, selectable by the user. The default method discretizes each attribute into equal-sized intervals, and has no effect on the asymptotic time complexity of the algorithm. Discretization can also be performed using a method similar to Catlett's (1991), repeatedly choosing the partition that minimizes entropy until one of several termination conditions is met. This causes learning time to become $O(e \log e)$, but may improve accuracy in some situations. Finally, numeric attributes can be handled directly by testing a condition of each type ($a_i > v_{ij}$ and $a_i < v_{ij}$) at each value v_{ij}. This does not change the asymptotic time complexity, but may cause v to become very large. Each of the last two methods may improve accuracy in some situations, at the cost of additional running time. However, uniform discretization is surprisingly robust (see (Dougherty, Kohavi & Sahami, 1995)), and can result in higher accuracy by helping to avoid overfitting.

Missing values are treated by letting them match any condition on the respective attribute, during both learning and classification.

Table 7.4. The optimized CWS algorithm.

Input: ES is the training set.

Procedure CWS (ES)

Let $RS = \emptyset$.
Let $Cscores(E) = 0$ for all E, C.
Repeat
  Add one active rule R_{new} with empty body to RS.
  Let $Cprobs(R_{new}) = 0$ for all C.
  For each active rule R in RS,
    For each example E covered by R,
      Subtract $Cprobs(R)$ from $Cscores(E)$.
    For each possible antecedent AV,
      Let $\Delta_{AV}Acc(RS) = 0$.
      Let $R_{AV} = R$ with AV conjoined to its body.
      Compute $Cprobs(R_{AV})$ and R_{AV}'s predicted class.
      For each example E_{AV} covered by R_{AV}
        Add $Cprobs(R_{AV})$ to $Cscores(E_{AV})$.
      For each example E covered by R
        Assign E to class with maximum $Cscore(E)$.
        Compute $\Delta_{AV}Acc_E(RS)$ (equal to -1/e, 0 or +1/e)
        Add $\Delta_{AV}Acc_E(RS)$ to $\Delta_{AV}Acc(RS)$.
    Pick AV' with maximum $\Delta_{AV}Acc(RS)$.
    If $\Delta_{AV'}Acc(RS) > 0$
      Then $R = R_{AV'}$,
      Else Deactivate R.
    For each example E covered by R ($R = R_{AV'}$ or not)
      Add $Cprobs(R)$ to $Cscores(E)$.
  If R_{new} has been deactivated
    Then Delete it.
Until all rules in RS are inactive.
Return RS.

7.5 Empirical Evaluation of CWS

CWS was empirically compared with C4.5RULES and CN2 along three variables: running time, accuracy, and comprehensibility of the output. All running times were obtained on a Sun 670 computer. Output size was taken as a rough measure of comprehensibility, counting one unit for each antecedent and consequent in each rule (including the default rule, with no antecedents and one consequent). This measure is imperfect for two reasons. First, for each system the meaning of a rule is not necessarily transparent: in CWS and CN2 overlapping rules are probabilistically combined to yield a class prediction, and in C4.5RULES each rule's antecedent side is implicitly conjoined with the negation of the antecedents of all preceding rules of different classes. Second, output simplicity is not the only factor in comprehensibility, which is ultimately subjective. However, it is an acceptable and frequently used approximation, especially when the systems being compared have similar output, as here (see (Catlett, 1991) for further discussion).

A preliminary study was conducted using the Boolean concept $abc \vee def$ as the learning target, with each disjunct having a probability of appearing in the data of 0.25, with 13 irrelevant attributes, and with 20% class noise (i.e., each class label has a probability of 0.2 of being flipped). Figure 7.5 shows the evolution of learning time with the number of examples for CWS and C4.5RULES, on a log-log scale. Canonical functions approximating each curve are also shown, as well as $e \log^2 e$, the running time observed by Cohen (Cohen, 1995) for RIPPERk and IREP. CWS's running time grows linearly with the number of examples, as expected, while C4.5RULES's is $O(e^2 \log e)$. CWS is also much faster than IREP and RIPPERk (note that, even though the log-log plot shown does not make this evident, the difference between e and $e \log^2 e$ is much larger than e).

CWS is also more accurate than C4.5RULES for each number of examples, converging to within 0.6% of the Bayes optimum (80%) for only 500 examples, and reaching it with 2500, while C4.5RULES never rises above 75%. CWS's output size stabilizes at 9, while C4.5RULES's increases from 17 for 100 examples to over 2300 for 10000. Without noise, both systems learn the concept easily. Thus these results indicate that CWS is more robust with respect to noise, at least in this simple domain. CN2's behavior is similar to C4.5RULES's in time and accuracy, but it produces larger rule sets.

The relationship between the theoretical bound of $O(eavcs)$ and CWS's actual average running time was investigated by running the system on 28 datasets from the UCI repository.[4] Figure 7.6 plots CPU time against the product $eavcs$.

[4] Audiology, annealing, breast cancer, credit, chess, Pima diabetes, echocardiogram, glass, heart disease, hepatitis, horse colic, thyroid disease, iris, labor, lung cancer, liver disease, lymphography, mushroom, post-operative, promoters, primary tumor, solar flare, sonar, soybean, splice junctions, voting, wine, and zoology; see Section 4.2. CWS tends to be less accurate than RISE on these datasets, implying that the latter is still preferable if accuracy is the main consideration and the dataset is not too large.

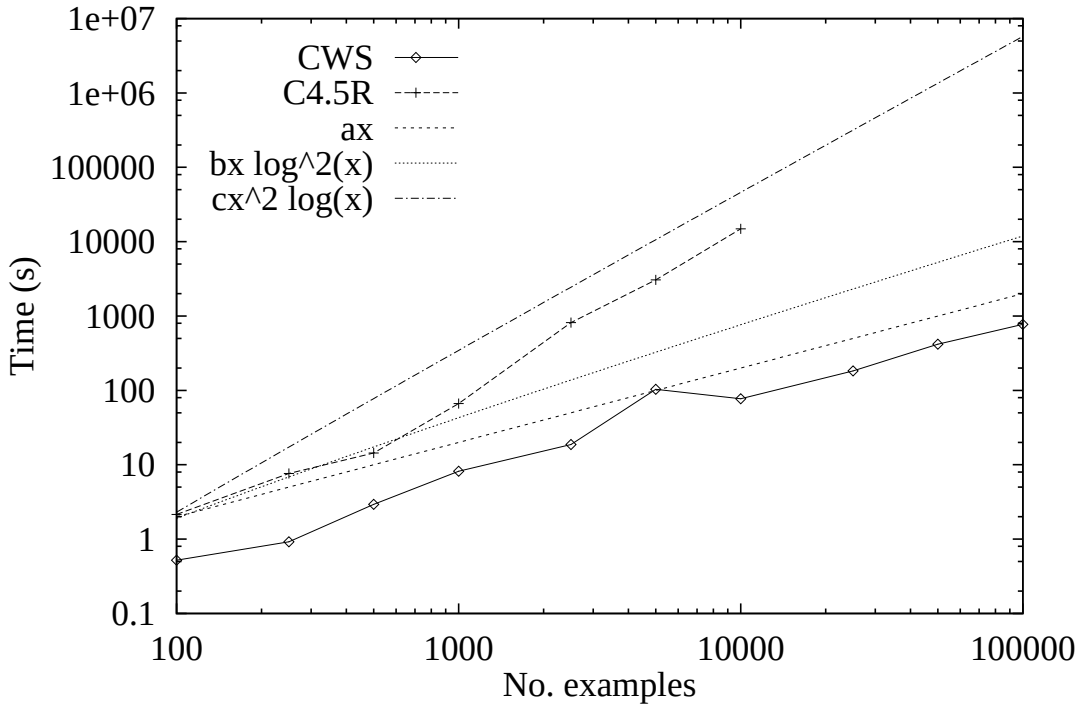

Figure 7.5. Learning times for the concept $abc \vee def$.

Linear regression yields the line $time = 1.1 \times 10^{-5}\ eavcs + 5.1$, with a correlation of 0.93 ($R^2 = 0.87$). Thus *eavcs* explains almost all the observed variation in CPU time, confirming the prediction of a linear bound.[5]

Experiments were also conducted on the space shuttle database with 20% class noise. The evolution of learning time with the number of training examples for CWS and C4.5RULES is shown in Figure 7.7 on a log-log scale, with approximate asymptotes also shown. CWS's curve is approximately log-linear, with the logarithmic factor attributable to the direct treatment of numeric values that was employed. (Uniform discretization resulted in linear time, but did not yield the requisite very high accuracies.) C4.5RULES's curve is roughly cubic. Extrapolating from it, C4.5RULES's learning time for the full database would be well over a month, while CWS takes 11 hours.

Learning curves are shown in Figure 7.8. CWS's accuracy is higher than C4.5RULES's for most points, and generally increases with the number of examples, showing that there is gain in using the larger samples, up to the full dataset. Figure 7.9 shows the evolution of output size. CWS's is low and almost constant, while C4.5RULES's grows to more than 500 by $e = 32000$. Up to 8000 examples, CN2's running time is similar to C4.5RULES's, but its output size grows to over

[5] Also, there is no correlation between the number of examples e and the output size s ($R^2 = 0.0004$).

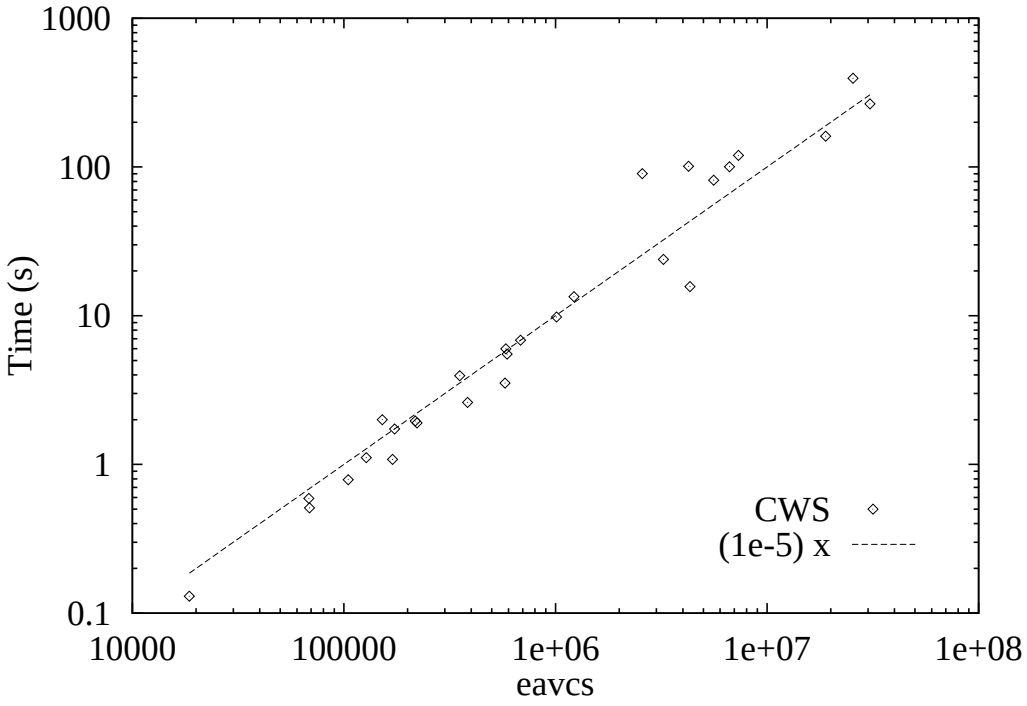

Figure 7.6. Relationship of empirical and theoretical learning times.

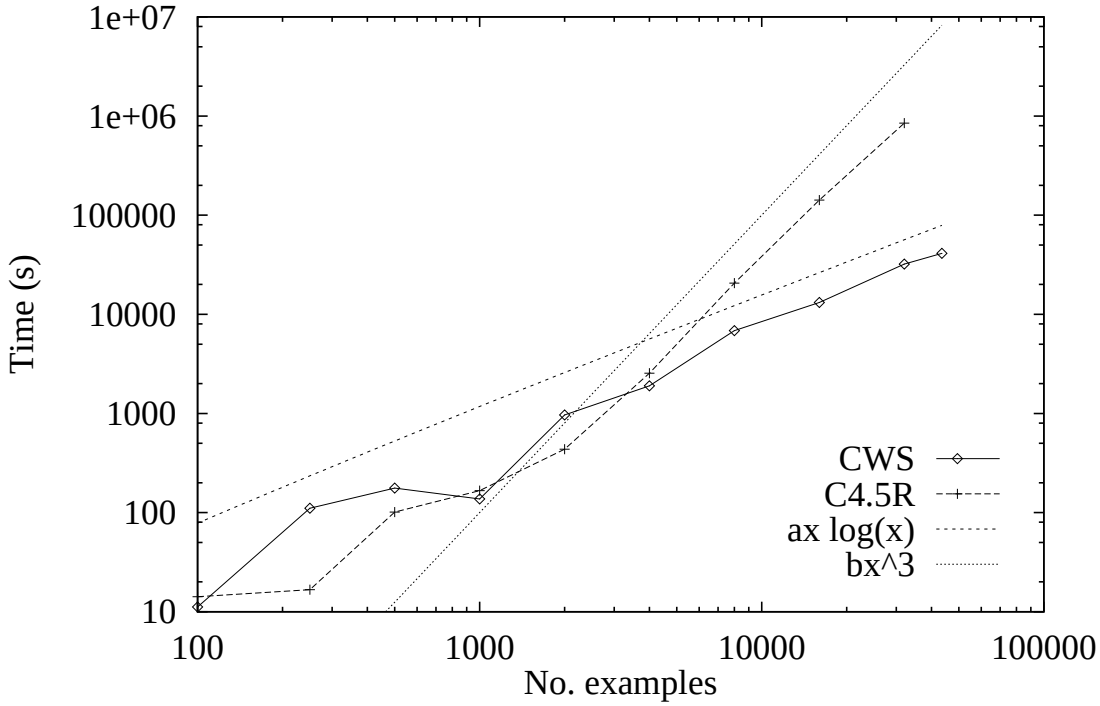

Figure 7.7. Learning times for the shuttle database.

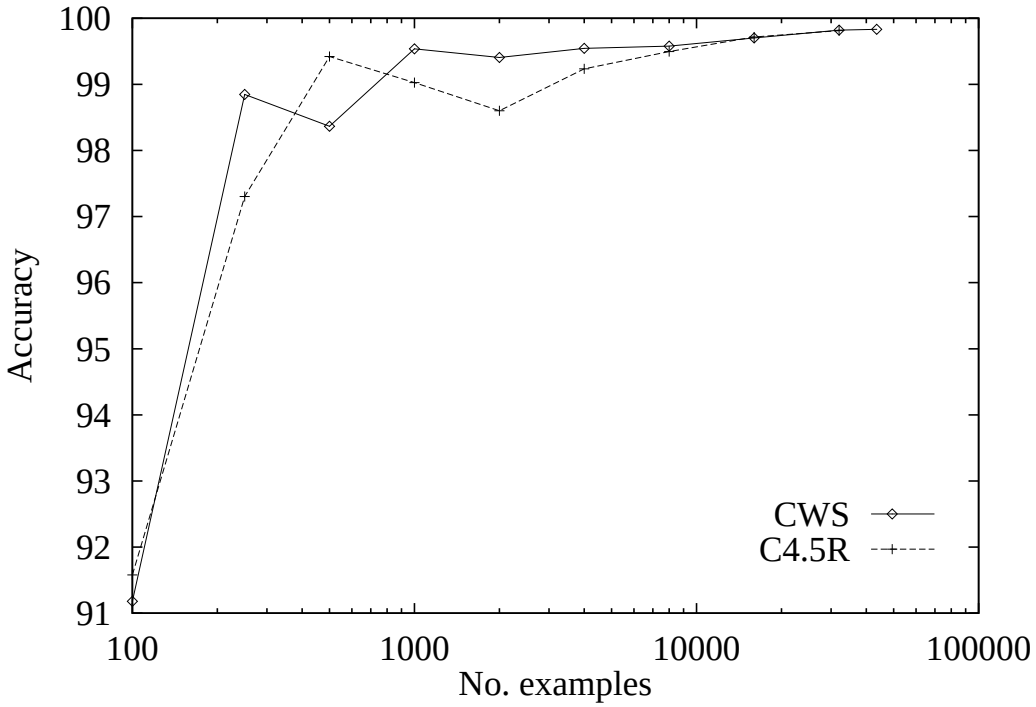

Figure 7.8. Learning curves for the shuttle database.

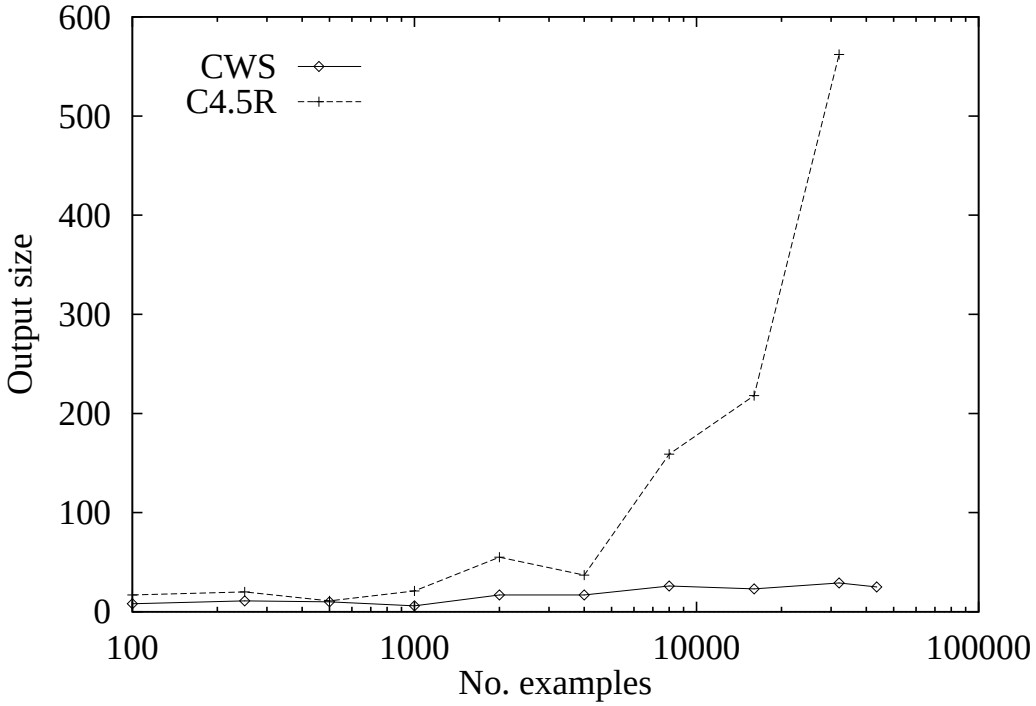

Figure 7.9. Output size growth for the shuttle database.

1700, and its accuracy never rises above 94%.[6] In summary, in this domain CWS outperforms C4.5RULES and CN2 in running time, accuracy and output size.

Compared to the noise-free case, the degradation in CWS's accuracy is less than 0.2% after $e = 100$, the rule set size is similar, and learning time is only degraded by a constant factor (of a few percent, on average). Thus CWS is again verified to be quite robust with respect to noise.

These experiments show that CWS holds promise as a data mining algorithm, able to produce fast, accurate and comprehensible results on large, noisy databases.

7.6 Summary

RISE can be applied to large databases through the use of sampling techniques like windowing and partitioning. In low-noise conditions, both of these methods are successful in reducing running time while maintaining accuracy, and partitioning sometimes improves accuracy significantly. In noisy conditions, the performance of windowing deteriorates markedly, while that of partitioning remains stable.

The ideal rule induction algorithm for data mining runs in linear or near-linear time, without compromising the accuracy or comprehensibility of the results. RISE's "conquering without separating" approach can be used to design such an algorithm. Empirical study shows that CWS, the result, can be used to advantage when the underlying concept is simple and the data is plentiful but noisy.

[6] For $e > 8000$ the program crashed due to lack of memory. This may be due to other jobs running concurrently.

Chapter 8
Conclusion

8.1 Contributions of this Dissertation

This dissertation presented a new approach to induction, unifying rule induction and instance-based learning, and addressing some of the problems of each approach by bringing in elements of the other. Its contributions include:

- *Simplification via unification of rule induction and IBL.* The simplification brought about by unifying two leading induction paradigms is of value to both the theorist and the practitioner. From a theoretical perspective, the unification of IBL and rule induction brings a degree of order and structure to the existing jungle of learning methods, overcomes some superficial distinctions while pointing out deeper ones, and opens up a spectrum of biases of which the previous algorithms are only special cases. From a practical perspective, the knowledge engineer's task is simplified, because a single algorithm can now be used instead of an IBL and a rule induction one, with reasonable confidence that accuracy will not be reduced, and may in fact be improved. In contrast to other empirical multistrategy learning approaches, the new algorithm is as simple as (or simpler than) the algorithms that it combines, and therefore as easy to understand, implement and apply.
- *Improved understanding of rule induction and IBL algorithms.* The framework presented here clarifies the relation between rule induction and IBL, and shows how rule and instance representations are related both syntactically and semantically. The experiments carried out in artificial domains show how concept specificity affects the behavior of rule induction algorithms, and how context dependency of feature relevance affects the behavior of IBL. Thus a better understanding of these two biases has been achieved.
- *Reduction of the fragmentation problem in rule induction.* RISE's "conquering without separating" approach to rule induction is able to largely circumvent the fragmentation problem that "separate and conquer" approaches suffer from. By letting each rule "see" the entire training set at each step of its induction, the incorrect decisions that result from looking at only a small fraction of the sample can be minimized. Although such an approach might

at first sight seem to lead to significantly increased computational costs, careful optimization makes it possible to apply it with the same asymptotic costs as other methods.

- *Reduction of the small disjuncts problem in rule induction.* While "separate and conquer" rule induction algorithms find small disjuncts both when they are really present and when they only appear to be as a result of fragmentation, RISE should be mainly affected only by the true, unavoidable small disjuncts. Studies in artificial domains varying the degree of specificity of the target disjuncts support this conclusion, showing that RISE is indeed at an advantage when learning small disjuncts.
- *First context-sensitive feature selection algorithm for IBL.* A significant limitation of classical methods for feature selection in IBL is that they do not take into account the fact that some features may be relevant in some parts of the instance space but not others, leading either to the deletion of important features, or to retaining features that act as noise when classifying some examples. RISE is able to select different features in different parts of the instance space, leading to significantly more correct frontiers in the many domains where such context effects are present. Studies in both benchmark and artificial domains showed this at work.
- *Accuracy improvements resulting from all of the above.* The immediate practical consequence of all the improvements above is that, in many domains, levels of predictive accuracy are achieved that were beyond the reach of previous rule induction and IBL methods. This was verified by extensive empirical studies, where, in approximately half of the domains studied, RISE not only matched but exceeded the accuracy of the best of its parent paradigms. Improved accuracy can translate into significant benefits in each of the application fields.
- *Accuracy improvements resulting from two-way induction.* Further improvements in predictive accuracy were obtained by combining RISE's specific-to-general search for rules with the more widely-used general-to-specific search direction. The combination of these two complementary biases is particularly suited to learning in domains where both general and specific rules are needed. These domains occur often in practice, making two-way induction a useful tool for the knowledge engineer.
- *First successful application of partitioning to specific-to-general rule induction.* RISE can be successfully applied to large databases through the use of the partitioning technique, making it a viable data mining algorithm, and extending its benefits to this type of application. By taking advantage of RISE's characteristics, partitioning can lead to simultaneous improvements in running time and accuracy using very simple combination methods. This contrasts with the more involved combination schemes that have been found necessary in general-to-specific approaches (Chan & Stolfo, 1995a).

- *First non-trivial linear-time rule induction algorithm.* Applying RISE's "conquering without separating" approach in a general-to-specific setting leads to an algorithm that, for the first time, runs in time linear in all the relevant parameters, while achieving accuracies at the level of more computationally expensive, state-of-the-art systems, and producing much simpler output. Its benefits were especially apparent in noisy domains. This combination of speed, accuracy, comprehensibility and robustness with respect to noise is potentially of great value in data mining applications.

8.2 Directions for Future Research

Any piece of research is only a node in a tree, and itself potentially the root of a new subtree of research. This section describes some possible new branches of the node represented by this dissertation.

One major area for future research concerns further study of RISE's bias. The experiments described in Chapters 4, 5 and 6 have only begun to explore this question. Ideally, we would like to arrive at a fairly complete picture of the assumptions that RISE incorporates, its strengths and weaknesses, the factors that influence its performance, and the conditions under which it can be expected to learn successfully. However, this is a large task—it has not been fully carried out for any machine learning algorithm, even much older ones than RISE. Even simple statistical pattern recognition algorithms that derive from very clear assumptions can have a range of applicability that is not trivially inferrable from those assumptions (e.g., (Domingos & Pazzani, 1996)). The method of choice for this type of research is to carry out studies in artificial domains, systematically varying relevant parameters, as was done in this dissertation. Analytical characterizations, where feasible, would also be useful; for example, it would be good to have an average-case analysis of RISE's learning time, to complement the worst-case one in Chapter 3.

One example of an aspect of RISE's bias that has not been fully studied is RISE's ability to form non-axis-parallel frontiers, while also dropping features where necessary. It would be interesting to investigate in detail how well this approach fares relative to axis-parallel ones, and relative to IBL using different types of similarity measure (Ting, 1994). Another example is RISE's handling of small disjuncts. This was investigated in this dissertation, but the picture is not yet complete. For example, RISE's advantage when attempting to learn a small disjunct may also derive from the fact that it tends to generate more versions of it in the same rule set than (say) CN2 or C4.5RULES, and thus may form a closer approximation of the theoretical Bayesian ideal (Buntine, 1990; Madigan, Raftery, Volinsky & Hoeting, 1996). This issue can be investigated empirically, and may lead to ideas on learning small disjuncts that are also useful in other rule learning contexts.

Another direction for research is extending the current RISE framework to include analytical as well as empirical learning. This will make it possible to perform theory revision and interleaved automatic/manual construction of knowledge bases, as well as pure induction. Several possible components for this are already in place: the basic RISE algorithm allows generalization of pre-existing rules as well as generalization from examples (since these examples are simply very specific cases of rules); because of RISE's best-match procedure, semantic specialization of rules is also possible, via syntactic generalization of competing rules and/or misclassified examples. The CWS algorithm allows direct specialization of existing rules, although its evaluation procedure and RISE's would need to be harmonized if the two algorithms were to be applied together. TWI also allows a form of theory revision: if an expert-supplied domain theory is used as the G component, RISE then plays the part of refining this theory by finding exceptions to it.

In two-way induction, more sophisticated methods of combining the two components may prove frutiful. Another potentially productive extension of TWI is inputting unpruned rules or trees, and combining the post-pruning stage with the specific-to-general induction process. Preliminary work has been done in this direction.

Viewing RISE as an instance-based learner, an obvious extension is to allow k-nearest-rule classification, by analogy with k-nearest neighbor. Unlike the latter, this requires changes in both learning and performance components, and these can be made in several ways. Experimentation should tell which are best when, and how useful they can be.

Beyond this, many further developments of RISE are possible. These include: methods for generalizing a symbolic antecedent that stop short of deleting it (e.g., through the introduction of internal disjunctions); alternative methods for processing missing values; methods for incorporating different misclassification costs and attribute measurement costs into RISE; post-processing methods for making RISE's output simpler and easier to understand, beyond those described in Chapter 4; and alternative distance measures, especially in domains where relevant knowledge can be incorporated into them.

Only a first pass at RISE's application to data mining problems was made here. Directions for future work include testing and developing more sophisticated methods of combining the outputs of the individual partitions (e.g., (Chan & Stolfo, 1995b)), automating the selection of partition size, and testing partitioning on a larger variety of larger databases. It may also be possible to further speed up RISE through the use of efficient data structures like k-d trees (Bentley, 1975; Friedman, Bentley & Finkel, 1977). Similarly, CWS admits to much further development. Directions for this include exploring ways of boosting its accuracy (or, conversely, broadening the set of concepts it can learn effectively) without affecting its asymptotic time complexity, and applying it to larger databases and problems in different areas. Work in this direction has also been initiated.

Another promising direction for research is to extend the RISE framework to perform regression. Several ways of doing this are readily apparent (e.g., each rule predicts the average value of the examples it covers, or forms a local linear regression model from those examples). RISE's ability to perform context-sensitive feature selection may be of great value in large-scale nonlinear regression problems, where a great many features are available, and different small subsets of these are relevant in different situations.

8.3 Summary

In the quest for knowledge, induction occupies center stage. Much of every human being's knowledge (and indeed, some of the deepest and hardest to formalize) is obtained by induction. Science proceeds largely by induction (even if not in as pure a form as was once commonly thought). In order to build knowledge-rich computational systems, induction from examples is often a better approach than attempting to make an expert's knowledge explicit, or is a key complement to this procedure. In today's information-rich world, induction is needed to extract general, relevant knowledge from the vast quantities of data available. The study of induction, the design of better algorithms for performing it and the investigation of their properties, can thus be of great value.

Concept learning is a key form of induction. This dissertation focused on two of the leading approaches to it: rule induction and instance-based learning. It identified some of the current weaknesses of each, and addressed them by bringing in elements of the other. In response to the fragmentation and small disjuncts problems that "separate and conquer" rule learners suffer from, a "conquering without separating" strategy was developed that relies on IBL's use of specific instances and similarity measures. In response to the context sensitivity problems of the classic feature selection algorithms used in IBL, an algorithm was developed that judges feature relevance in context, relying on rule learners' ability to select different features in different regions of the instance space. Extensive empirical study showed the resulting algorithm to be remarkably successful both as rule induction and as IBL, and demonstrated the usefulness of its ideas in learning from large databases.

Thus, it is hoped that this dissertation will constitute another step in our attempt to better understand and carry out induction, and, by extension, in the quest for knowledge of which induction is a prominent part.

Appendix A
UCI Databases

This appendix lists the UCI repository file containing each database, together with any conversions performed and any other necessary information. Instance identification codes were deleted wherever present. The UCI repository is accessible by anonymous ftp from ftp.ics.uci.edu, subdirectory pub/machine-learning-databases, and on the World Wide Web at the URL http://www.ics.uci.edu/-mlearn/MLRepository.html. Please see the documentation in the repository for further details on each of these databases.

AD: audiology/audiology.standardized.data

AN: annealing/anneal.data

BC: breast-cancer/breast-cancer.data

CE: credit-screening/crx.data

CH: chess/king-rook-vs-king-pawn/kr-vs-kp.data

DI: pima-indians-diabetes/pima-indians-diabetes.data

EC: echocardiogram/echocardiogram.data
Class is second attribute, attributes 1 and 10-13 deleted, example with unknown class deleted.

GL: glass/glass.data

HD: heart-disease/cleve.mod
Last attribute deleted to yield a two-class problem.

HE: hepatitis/hepatitis.data

HO: horse-colic/horse-colic.data
Class is 24th attribute, attributes 3 and 25-28 deleted.

HY: thyroid-disease/hypothyroid.data

IR: iris/iris.data

LA: Quinlan's formatting of this dataset, distributed with C4.5, was used instead of the less standard one in the UCI repository.

LC: lung-cancer/lung-cancer.data

LD: liver-disorders/bupa.data

LE: lenses/lenses.data

LI: led-display-creator/led-creator.c
Program run with the following parameters: 100 examples, seed = 1, 10% noise.

LY: lymphography/lymphography.data

MU: mushroom/agaricus-lepiota.data

PO: postoperative-patient-data/post-operative.data
Pseudo-discretized values converted to numeric (e.g., high, mid, and low become 1, 0 and -1).

PR: molecular-biology/promoter-gene-sequences/promoters.data

PT: primary-tumor/primary-tumor.data

SF: solar-flare/flare.data1
First attribute used as class.

SN: undocumented/connectionist-bench/sonar/sonar.all-data

SO: soybean/soybean-small.data

SP: molecular-biology/splice-junction-gene-sequences/splice.data

VO: voting-records/house-votes-84.data

WI: wine/wine.data

ZO: zoo/zoo.data

Appendix B
Creation of Prototypes

This appendix describes how, for each one of p prototypes, the relevant features are chosen at random in a way that guarantees that the feature difference between the prototypes (Equation 6.1) is on average a pre-specified value D. The use of these prototypes is described in Section 6.5.

If p_k is the number of prototypes in which feature k is relevant, Equation 6.1 can also be written as:

$$D = \frac{2}{p(p-1)} \sum_{k=1}^{a} p_k(p - p_k) \tag{B.1}$$

where p corresponds to e in (6.1), and a is the number of features. Let:

$$\pi_k = p_k(p - p_k) \tag{B.2}$$

and let $\overline{\pi}$ be the average value of π for the a features. $\overline{\pi}$ is determined by the desired value of D:

$$\overline{\pi} = \frac{p(p-1)}{2a} D \tag{B.3}$$

Next, k values of π_k such that their average is the value $\overline{\pi}$ above can be obtained from a uniform distribution in the interval $[0, 2\overline{\pi}]$. The corresponding p_ks are found by solving (B.2) for p_k, which is possible in general iff:

$$D \leq \frac{a}{4(1 - \frac{1}{p})} \tag{B.4}$$

This constrains the observable range of D given a and p. Finally, feature k is included in each prototype with probability p_k/p.

Appendix C
Bayesian Averaging of RISE Rule Sets

This appendix describes the use of Bayesian model averaging (Madigan, Raftery, Volinsky & Hoeting, 1996) to combine the rule sets produced by partitioning a database and running RISE on each partition separately (Section 7.3).

Given an individual partition, let e_{max} be the partition size, $\vec{x}$ the examples in that partition, $\vec{c}$ the corresponding class labels, and h the rule set produced. Then, by Bayes's theorem, and assuming the examples are drawn independently:

$$Pr(h|\vec{x},\vec{c}) = \frac{Pr(h)}{Pr(\vec{x},\vec{c})} \prod_{i=1}^{e_{max}} Pr(x_i, c_i|h) \tag{C.1}$$

where the data prior $Pr(\vec{x},\vec{c})$ is the same for all models, and can be ignored.

$Pr(h)$ is the prior probability of h, and is assumed uniform (i.e., Dirichlet with parameter $\alpha = 1$). For each pair (x_i, c_i) in the partition, $Pr(x_i, c_i|h)$ is computed as the probability of an example having class c_i given that it is in the region won by the rule that wins x_i. This probability is estimated empirically from the examples won by that rule. Let r be this rule, e_r the total number of examples it wins, and e_{r,c_i} the number of examples of class c_i that it wins. Then:

$$\hat{Pr}(x_i, c_i|h) = \frac{e_{r,c_i}}{e_r} \tag{C.2}$$

This is analogous to the treatment in (Buntine, 1990), using the partition of the instance space induced by the rules in the same way Buntine uses the partition induced by a decision tree. Finally, if H is the set of all rule sets h induced from the partitions, a test example x is assigned to the class that maximizes:

$$Pr(c|x,H) = \sum_{h \in H} Pr(c|x,h)\, Pr(h|\vec{x},\vec{c}) \tag{C.3}$$

Note that each rule set is induced from a different partition, so, strictly speaking, $\vec{x}$, $\vec{c}$ and their components should be indexed by h. This was omitted for the sake of simplicity.

References

Agresti, A. (1990). *Categorical Data Analysis.* New York, NY: Wiley.

Aha, D. W. (1989). Incremental, instance-based learning of independent and graded concept descriptions. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 387–391). Ithaca, NY: Morgan Kaufmann.

Aha, D. W. (Ed.) (1997). Special issue on lazy learning. *Artificial Intelligence Review, 11.*

Aha, D. W., & Bankert, R. L. (1994). Feature selection for case-based classification of cloud types: An empirical comparison. *Proceedings of the AAAI-94 Workshop on Case-Based Reasoning* (pp. 106–112). Seattle, WA: AAAI Press.

Aha, D. W., & Goldstone, R. L. (1992). Concept learning and flexible weighting. *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society* (pp. 534–539). Evanston, IL: Lawrence Erlbaum.

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

Ali, K., & Pazzani, M. (1996). Classification using Bayes averaging of multiple, relational rule-based models. In D. Fisher & H.-J. Lenz (Eds.), *Learning from Data: Artificial Intelligence and Statistics V* (pp. 207–217). New York, NY: Springer-Verlag.

Almuallim, H., & Dietterich, T. G. (1991). Learning with many irrelevant features. *Proceedings of the Ninth National Conference on Artificial Intelligence* (pp. 547–552). Menlo Park, CA: AAAI Press.

Anderson, J. A., & Rosenfeld, E. (Eds.) (1988). *Neurocomputing: Foundations of Research.* Cambridge, MA: MIT Press.

Atkeson, C. G., Moore, A. W., & Schaal, S. (1997). Locally weighted learning. *Artificial Intelligence Review, 11,* 11–73.

Belew, R. K., McInerney, J., & Schraudolph, N. N. (1992). Evolving networks: Using the genetic algorithm with connectionist learning. In C. G. Langton, C. Taylor, J. D. Farmer, & S. Rasmussen (Eds.), *Artificial Life II* (pp. 511–547). Redwood City, CA: Addison-Wesley.

Bellman, R. E. (1961). *Adaptive Control Processes: A Guided Tour.* Princeton, NJ: Princeton University Press.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, *18*, 509–517.

Biberman, Y. (1994). A context similarity measure. *Proceedings of the Ninth European Conference on Machine Learning* (pp. 49–63). Catania, Italy: Springer-Verlag.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. K. (1987). Occam's razor. *Information Processing Letters*, *24*, 377–380.

Booker, L. B., Goldberg, D. E., & Holland, J. H. (1989). Classifier systems and genetic algorithms. *Artificial Intelligence*, *40*, 235–282.

Breiman, L. (1996a). Bagging predictors. *Machine Learning*, *24*, 123–140.

Breiman, L. (1996b). Bias, variance and arcing classifiers. Technical Report 460, Statistics Department, University of California at Berkeley, Berkeley, CA. Available electronically as ftp://ftp.stat.berkeley.edu/users/breiman/-arcall.ps.Z.

Breiman, L. (1996c). Pasting bites together for prediction in large data sets and on-line. Technical report, Statistics Department, University of California at Berkeley, Berkeley, CA. Available electronically as ftp://ftp.stat.berkeley.-edu/users/breiman/pastebite.ps.

Breiman, L. (1996d). Stacked regressions. *Machine Learning*, *24*, 49–64.

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

Brodley, C. E. (1995). Recursive automatic bias selection for classifier construction. *Machine Learning*, *20*, 63–94.

Brunk, C., & Pazzani, M. J. (1991). An investigation of noise-tolerant relational concept learning algorithms. *Proceedings of the Eighth International Workshop on Machine Learning* (pp. 389–393). Evanston, IL: Morgan Kaufmann.

Buntine, W., & Caruana, R. (1992). Introduction to IND version 2.1 and recursive partitioning. Technical report, NASA Ames Research Center, Moffett Field, CA.

Buntine, W. L. (1990). *A Theory of Learning Classification Rules*. PhD thesis, School of Computing Science, University of Technology, Sydney, Australia.

Cain, T., Pazzani, M. J., & Silverstein, G. (1991). Using domain knowledge to influence similarity judgments. *Proceedings of the Case-Based Reasoning Workshop* (pp. 191–199). Washington, DC: Morgan Kaufmann.

Cameron-Jones, R. M. (1992). Minimum description length instance-based learning. *Proceedings of the Fifth Australian Joint Conference on Artificial Intelligence* (pp. 368–373). Hobart, Australia: World Scientific.

Cardie, C. (1993). Using decision trees to improve case-based learning. *Proceedings of the Tenth International Conference on Machine Learning* (pp. 25–32). Amherst, MA: Morgan Kaufmann.

Caruana, R., & Freitag, D. (1994). Greedy attribute selection. *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 28–36). New Brunswick, NJ: Morgan Kaufmann.

Catlett, J. (1991). *Megainduction: Machine Learning on Very Large Databases.* PhD thesis, Basser Department of Computer Science, University of Sydney, Sydney, Australia.

Chan, P. K., & Stolfo, S. J. (1995a). A comparative evaluation of voting and meta-learning on partitioned data. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 90–98). Tahoe City, CA: Morgan Kaufmann.

Chan, P. K., & Stolfo, S. J. (1995b). Learning arbiter and combiner trees from partitioned data for scaling machine learning. *Proceedings of the First International Conference on Knowledge Discovery and Data Mining* (pp. 39–44). Montréal, Canada: AAAI Press.

Cheeseman, P. (1990). On finding the most probable model. In J. Shrager & P. Langley (Eds.), *Computational Models of Scientific Discovery and Theory Formation* (pp. 73–95). San Mateo, CA: Morgan Kaufmann.

Clark, P., & Boswell, R. (1991). Rule induction with CN2: Some recent improvements. *Proceedings of the Sixth European Working Session on Learning* (pp. 151–163). Porto, Portugal: Springer-Verlag.

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

Clearwater, S., and Provost, F. (1990). RL4: A tool for knowledge-based induction. *Proceedings of the Second IEEE International Conference on Tools for Artificial Intelligence* (pp. 24–30). San Jose, CA: IEEE Computer Society Press.

Cohen, W. W. (1993). Efficient pruning methods for separate-and-conquer rule learning systems. *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence* (pp. 988–994). Chambéry, France: Morgan Kaufmann.

Cohen, W. W. (1995). Fast effective rule induction. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 115–123). Tahoe City, CA: Morgan Kaufmann.

Cortes, C. (1995). *Prediction of Generalization Ability in Learning Machines.* PhD thesis, Department of Computer Science, University of Rochester, Rochester, NY.

Cost, S., & Salzberg, S. (1993). A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, *10*, 57–78.

Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13*, 21–27.

Creecy, R. H., Masand, B. M., Smith, S. J., & Waltz, D. L. (1992). Trading MIPS and memory for knowledge engineering. *Communications of the ACM*, *35*(8), 48–63.

Dasarathy, B. W. (Ed.) (1991). *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. Los Alamitos, CA: IEEE Computer Society Press.

DeGroot, M. H. (1986). *Probability and Statistics* (2nd ed.). Reading, MA: Addison-Wesley.

Devijver, P. A., & Kittler, J. (1982). *Pattern Recognition: A Statistical Approach*. Englewood Cliffs, N.J.: Prentice-Hall.

Dietterich, T. (1996). Statistical tests for comparing supervised classification learning algorithms. Technical report, Department of Computer Science, Oregon State University, Corvallis, OR. Available electronically as ftp://ftp.cs.orst.edu/pub/tgd/papers/stats.ps.gz.

Domingos, P. (1994a). Design and evaluation of the RISE 1.0 system. Technical Report 94-34, Department of Information and Computer Science, University of California at Irvine, Irvine, CA.

Domingos, P. (1994b). The RISE system: Conquering without separating. *Proceedings of the Sixth IEEE International Conference on Tools with Artificial Intelligence* (pp. 704–707). New Orleans, LA: IEEE Computer Society Press.

Domingos, P. (1995). The RISE 2.0 system: A case study in multistrategy learning. Technical Report 95-2, Department of Information and Computer Science, University of California at Irvine, Irvine, CA.

Domingos, P. (1997a). Bayesian model averaging in rule induction. *Preliminary Papers of the Sixth International Workshop on Artificial Intelligence and Statistics* (pp. 157–164). Fort Lauderdale, FL: Society for Artificial Intelligence and Statistics.

Domingos, P. (1997b). Why does bagging work? A Bayesian account and its implications. *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining* (to appear). Newport Beach, CA: AAAI Press.

Domingos, P., & Pazzani, M. (1996). Beyond independence: Conditions for the optimality of the simple Bayesian classifier. *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 105–112). Bari, Italy: Morgan Kaufmann.

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 194–202). Tahoe City, CA: Morgan Kaufmann.

Draper, N. R., & Smith, H. (1981). *Applied Regression Analysis* (2nd ed.). New York, NY: Wiley.

Duda, R. O., & Hart, P. E. (1973). *Pattern Classification and Scene Analysis.* New York, NY: Wiley.

Everitt, B. (1980). *Cluster Analysis* (2nd ed.). London, UK: Heinemann.

Fayyad, U. M., & Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence* (pp. 1022–1027). Chambéry, France: Morgan Kaufmann.

Fayyad, U. M., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery: An overview. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, & R. Uthurusamy (Eds.), *Advances in Knowledge Discovery and Data Mining* (pp. 1–34). Menlo Park, CA: AAAI Press.

Fayyad, U. M., & Uthurusamy, R. (Eds.) (1995). *Proceedings of the First International Conference on Knowledge Discovery and Data Mining.* Montréal, Canada: AAAI Press.

Fisher, D. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, *2*, 139–172.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 148–156). Bari, Italy: Morgan Kaufmann.

Friedman, J. H. (1996). On bias, variance, 0/1 - loss, and the curse-of-dimensionality. Technical report, Department of Statistics and Stanford Linear Accelerator Center, Stanford University, Stanford, CA. Available electronically as ftp://playfair.stanford.edu/pub/friedman/kdd.ps.Z.

Friedman, J. H., Bentley, J. L., & Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, *3*, 209–226.

Fürnkranz, J., & Widmer, G. (1994). Incremental reduced error pruning. *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 70–77). New Brunswick, NJ: Morgan Kaufmann.

Golding, A. R., & Rosenbloom, P. S. (1991). Improving rule-based systems through case-based reasoning. *Proceedings of the Ninth National Conference on Artificial Intelligence* (pp. 22–27). Menlo Park, CA: AAAI Press.

Good, I. J. (1965). *The Estimation of Probabilities: An Essay on Modern Bayesian Methods.* Cambridge, MA: MIT Press.

Goodman, R. M., Higgins, C. M., Miller, J. W., & Smyth, P. (1992). Rule-based neural networks for classification and probability estimation. *Neural Computation*, *4*, 781–804.

Gordon, L., & Olshen, R. A. (1984). Almost sure consistent nonparametric regression from recursive partitioning schemes. *Journal of Multivariate Analysis*, *15*, 147–163.

Hastie, T., & Tibshirani, R. (1996). Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *18*, 607–616.

Heckerman, D. (1996). Bayesian networks for knowledge discovery. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, & R. Uthurusamy (Eds.), *Advances in Knowledge Discovery and Data Mining* (pp. 273–305). Menlo Park, CA: AAAI Press.

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

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

John, G. H., Kohavi, R., & Pfleger, K. (1994). Irrelevant features and the subset selection problem. *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 121–129). New Brunswick, NJ: Morgan Kaufmann.

Kelly, J. D., & Davis, L. (1991). A hybrid genetic algorithm for classification. *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence* (pp. 645–650). Sydney, Australia: Morgan Kaufmann.

Kibler, D., & Aha, D. W. (1987). Learning representative exemplars of concepts: An initial case study. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 24–30). Irvine, CA: Morgan Kaufmann.

Kira, K., & Rendell, L. A. (1992). A practical approach to feature selection. *Proceedings of the Ninth International Workshop on Machine Learning* (pp. 249–256). Aberdeen, Scotland: Morgan Kaufmann.

Kittler, J. (1986). Feature selection and extraction. In T. Y. Young & K. S. Fu (Eds.), *Handbook of Pattern Recognition and Image Processing*. New York, NY: Academic Press.

Kohavi, R. (1996). Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (pp. 202–207). Portland, OR: AAAI Press.

Kohavi, R., & Li, C. (1995). Oblivious decision trees, graphs, and top-down pruning. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* (pp. 1071–1077). Montréal, Canada: Morgan Kaufmann.

Kohavi, R., & Wolpert, D. H. (1996). Bias plus variance decomposition for zero-one loss functions. *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 275–283). Bari, Italy: Morgan Kaufmann.

Kolodner, J. (1993). *Case-Based Reasoning*. San Mateo, CA: Morgan Kaufmann.

Kong, E. B., & Dietterich, T. G. (1995). Error-correcting output coding corrects bias and variance. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 313–321). Tahoe City, CA: Morgan Kaufmann.

Langley, P., & Sage, S. (1994). Oblivious decision trees and abstract cases. *Proceedings of the AAAI-94 Workshop on Case-Based Reasoning* (pp. 113–117). Seattle, CA: AAAI Press.

Lee, C. (1994). An instance-based learning method for databases: An information theoretic approach. *Proceedings of the Ninth European Conference on Machine Learning* (pp. 387–390). Catania, Italy: Springer-Verlag.

Lowe, D. G. (1995). Similarity metric learning for a variable-kernel classifier. *Neural Computation*, *7*, 72–85.

Madigan, D., Raftery, A. E., Volinsky, C. T., & Hoeting, J. A. (1996). Bayesian model averaging. *Proceedings of the AAAI-96 Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms* (pp. 77–83). Portland, OR: AAAI Press.

Merz, C. J., Murphy, P. M., & Aha, D. W. (1997). UCI repository of machine learning databases. Machine-readable data repository, Department of Information and Computer Science, University of California at Irvine, Irvine, CA.

Michalski, R. S. (1983). A theory and methodology of inductive learning. *Artificial Intelligence*, *20*, 111–161.

Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (Eds.) (1983). *Machine Learning: An Artificial Intelligence Approach.* Palo Alto, CA: Tioga.

Michalski, R. S., Mozetic, I., Hong, J., & Lavrac, N. (1986). The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 1041–1045). Philadelphia, PA: AAAI Press.

Michalski, R. S., & Tecuci, G. (Eds.) (1993). *Proceedings of the Second International Workshop on Multistrategy Learning.* Harpers Ferry, VA: Office of Naval Research/George Mason University.

Michalski, R. S, & Tecuci, G. (Eds.) (1994). *Machine Learning: A Multistrategy Approach.* San Mateo, CA: Morgan Kaufmann.

Michalski, R. S., & Wnek, J. (Eds.) (1996). *Proceedings of the Third International Workshop on Multistrategy Learning.* Harpers Ferry, VA: AAAI Press.

Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (Eds.) (1994). *Machine Learning, Neural and Statistical Classification.* New York, NY: Ellis Horwood.

Minton, S. (1990). Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, *42*, 363–392.

Mitchell, T. M. (1980). The need for biases in learning generalizations. Technical report, Rutgers University, Computer Science Department, New Brunswick, NJ.

Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence*, *18*, 203–226.

Mitchell, T. M., Keller, R. M., & Kedar-Cabelli, S. T. (1986). Explanation-based generalization: A unifying view. *Machine Learning*, *1*, 47–80.

Mohri, T., & Tanaka, H. (1994). An optimal weighting criterion of case indexing for both numeric and symbolic attributes. *Proceedings of the AAAI-94 Workshop on Case-Based Reasoning* (pp. 123–127). Seattle, WA: AAAI Press.

Moody, J., & Darken, C. J. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*, *1*, 281–294.

Moore, A. W., & Lee, M. S. (1994). Efficient algorithms for minimizing cross validation error. *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 190–198). New Brunswick, NJ: Morgan Kaufmann.

Muggleton, S., & Feng, C. (1990). Efficient induction of logic programs. *Proceedings of the First Workshop on Algorithmic Learning Theory* (pp. 368–381). Tokyo, Japan: Springer-Verlag.

Murphy, P., & Pazzani, M. (1994). Exploring the decision forest: An empirical investigation of Occam's razor in decision tree induction. *Journal of Artificial Intelligence Research*, *1*, 257–275.

Nakhaeizadeh, G. (1995). Panel session abstract. *KDD-95 Panel on Commercial Knowledge Discovery Applications*. Montréal, Canada.

Niblett, T. (1987). Constructing decision trees in noisy domains. *Proceedings of the Second European Working Session on Learning* (pp. 67–78). Bled, Yugoslavia: Sigma.

Nosofsky, R. M., Clark, S. E., & Shin, H. J. (1989). Rules and exemplars in categorization, identification, and recognition. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *15*, 282–304.

Oliveira, A. L., & Sangiovanni-Vincentelli, A. (1995). Inferring reduced ordered decision graphs of minimum description length. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 421–429). Tahoe City, CA: Morgan Kaufmann.

Ourston, D., & Mooney, R. J. (1994). Theory refinement combining analytical and empirical methods. *Artificial Intelligence*, *66*, 273–309.

Pagallo, G., & Haussler, D. (1990). Boolean feature discovery in empirical learning. *Machine Learning*, *3*, 71–99.

Pazzani, M., & Kibler, D. (1992). The utility of knowledge in inductive learning. *Machine Learning*, *9*, 57–94.

Piatetsky-Shapiro, G. (1991). Discovery, analysis, and presentation of strong rules. In G. Piatetsky-Shapiro & W. J. Frawley (Eds.), *Knowledge Discovery in Databases*. Menlo Park, CA: AAAI Press.

Poggio, T., & Girosi, F. (1990). Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, *247*, 978–982.

Porter, B. W., Bareiss, R., & Holte, R. C. (1990). Knowledge acquisition and heuristic classification in weak-theory domains. *Artificial Intelligence*, *45*, 229–263.

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

Quinlan, J. R. (1987a). Generating production rules from decision trees. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (pp. 304–307). Milan, Italy: Morgan Kaufmann.

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

Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine Learning*, *5*, 239–266.

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

Quinlan, J. R. (1993b). Combining instance-based and model-based learning. *Proceedings of the Tenth International Conference on Machine Learning* (pp. 236–243). Amherst, MA: Morgan Kaufmann.

Quinlan, J. R., & Cameron-Jones, R. M. (1995). Oversearching and layered search in empirical learning. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* (pp. 1019–1024). Montréal, Canada: Morgan Kaufmann.

Ram, A. (1993). Indexing, elaboration, and refinement: Incremental learning of explanatory cases. *Machine Learning*, *10*, 201–248.

Rao, R. B., Gordon, D., & Spears, W. (1995). For every action, is there really an equal and opposite reaction? Analysis of the conservation law for generalization performance. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 471–479). Tahoe City, CA: Morgan Kaufmann.

Rendell, L. (1986). A general framework for induction and a study of selective induction. *Machine Learning*, *1*, 177–226.

Ricci, F., & Avesani, P. (1995). Learning a local similarity metric for case-based reasoning. *Proceedings of the First International Conference on Case-Based Reasoning* (pp. 301–312). Sesimbra, Portugal: Springer-Verlag.

Riesbeck, C. K., & Schank, R. C. (1989). *Inside Case-Based Reasoning*. Hillsdale, NJ: Lawrence Erlbaum.

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks*. Cambridge, UK: Cambridge University Press.

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

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1 (pp. 318–362). Cambridge, MA: MIT Press.

Rymon, R. (1993). An SE-Tree based characterization of the induction problem. *Proceedings of the Tenth International Conference on Machine Learning* (pp. 268–275). Amherst, MA: Morgan Kaufmann.

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

Schaffer, C. (1994a). Cross-validation, stacking, and bi-level stacking: Meta-methods for classification learning. In P. Cheeseman & R. W. Oldford (Eds.), *Selecting Models from Data: Artificial Intelligence and Statistics IV*. New York, NY: Springer-Verlag.

Schaffer, C. (1994b). A conservation law for generalization performance. *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 259–265). New Brunswick, NJ: Morgan Kaufmann.

Schlimmer, J. C. (1993). Efficiently inducing determinations: A complete and systematic search algorithm that uses optimal pruning. *Proceedings of the Tenth International Conference on Machine Learning* (pp. 284–290). Amherst, MA: Morgan Kaufmann.

Scott, P. D., & Sage, K. H. (1992). Why generalize? Hybrid representations and instance-based learning. *Proceedings of the Tenth European Conference on Artificial Intelligence* (pp. 484–486). Vienna, Austria: Wiley.

Segal, R., & Etzioni, O. (1994). Learning decision lists using homogeneous rules. *Proceedings of the Twelfth National Conference on Artificial Intelligence* (pp. 619–625). Seattle, WA: AAAI Press.

Simon, H. A. (1983). Why should machines learn? In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (pp. 25–37). Palo Alto, CA: Tioga.

Simoudis, E., Han, J., & Fayyad, U. (Eds.) (1996). *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. Portland, OR: AAAI Press.

Skalak, D. B. (1992). Representing cases as knowledge sources that apply local similarity metrics. *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society* (pp. 325–330). Evanston, IL: Lawrence Erlbaum.

Skalak, D. B. (1994). Prototype and feature selection by sampling and random mutation hill climbing algorithms. *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 293–301). New Brunswick, NJ: Morgan Kaufmann.

Smyth, P., Gray, A., & Fayyad, U. (1995). Retrofitting decision tree classifiers using kernel density estimation. *Proceedings of the Twelfth International Conference on Machine Learning*, Tahoe City, CA: Morgan Kaufmann.

Stanfill, C., & Waltz, D. (1986). Toward memory-based reasoning. *Communications of the ACM*, *29*, 1213–1228.

Sutton, R. S. (Ed.) (1992). *Reinforcement Learning.* Boston, MA: Kluwer.

Tibshirani, R. (1996). Bias, variance and prediction error for classification rules. Technical report, Department of Preventive Medicine and Biostatistics and Department of Statistics, University of Toronto, Toronto, Canada. Available electronically as http://utstat.toronto.edu/reports/tibs/biasvar.ps.

Ting, K. M. (1994). Discretization of continuous-valued attributes and instance-based learning. Technical Report 491, Basser Department of Computer Science, University of Sydney, Sydney, Australia.

Towell, G. G., & Shavlik, J. W. (1994). Knowledge-based artificial neural networks. *Artificial Intelligence*, *70*, 119–165.

Tumer, K., & Ghosh, J. (1996). Classifier combining: Analytical results and implications. *Proceedings of the AAAI-96 Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms* (pp. 126–132). Portland, OR: AAAI Press.

Utgoff, P. E. (1989a). Perceptron trees: A case study in hybrid concept representations. *Connection Science*, *1*, 377–391.

Utgoff, P. E. (1989b). Incremental induction of decision trees. *Machine Learning*, *4*, 161–186.

Vafaie, H., & DeJong, K. (1993). Robust feature selection algorithms. *Proceedings of the Fifth IEEE International Conference on Tools for Artificial Intelligence* (pp. 356–363). Boston, MA: IEEE Computer Society Press.

Webb, G. I. (1995). OPUS: An efficient admissible algorithm for unordered search. *Journal of Artificial Intelligence Research*, *3*, 431–465.

Webb, G. I. (1996). Further experimental evidence against the utility of Occam's razor. *Journal of Artificial Intelligence Research*, *4*, 397–417.

Weiss, S. M., Galen, R. M., & Tadepalli, P. V. (1987). Optimizing the predictive value of diagnostic decision rules. *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 521–526). Seattle, WA: AAAI Press.

Weiss, S. M., & Indurkhya, N. (1995). Rule-based machine learning methods for functional prediction. *Journal of Artificial Intelligence Research*, *3*, 383–403.

Wettschereck, D. (1994). A hybrid nearest-neighbor and nearest-hyperrectangle algorithm. *Proceedings of the Ninth European Conference on Machine Learning* (pp. 323–335). Catania, Italy: Springer-Verlag.

Wettschereck, D., & Dietterich, T. (1995). An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. *Machine Learning*, *19*, 5–27.

Widrow, B., & Stearns, S. D. (1985). *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall.

Wnek, J., & Michalski, R. S. (1994). Hypothesis-driven constructive induction in AQ17-HCI: A method and experiments. *Machine Learning*, *14*, 139–168.

Wolpert, D. (1992). Stacked generalization. *Neural Networks*, *5*, 241–259.

Wolpert, D. (1996). The lack of a priori distinctions between learning algorithms. *Neural Computation*, *8*, 1341–1390.

Zhang, J. (1990). A method that combines inductive learning with exemplar-based learning. *Proceedings of the Second IEEE International Conference on Tools for Artificial Intelligence* (pp. 31–37). San Jose, CA: IEEE Computer Society Press.