A Study of Instance-Based Algorithms for Supervised Learning Tasks:

Mathematical, Empirical, and Psychological Evaluations

David W. Aha

Department of Information and Computer Science
University of California, Irvine, CA 92717

Technical Report 90-42

November 27, 1990

Dissertation
submitted in partial satisfaction of the requirements for the degree of
Doctor of Philosophy in Information and Computer Science

Dedication

This dissertation is dedicated to my parents,
Robert S. and Margaret E. Aha, who supported
and followed my progress with pride and love.

This dissertation is also dedicated to
Stephanie Hardie Leif Aha, my partner in life,
who shared my experiences, both good and bad,
and patiently tolerated my obsessions.

Contents

Acknowledgements

Aha, now that does sound promising!
– Douglas Adams (1986, page 135)

I'll never forget the first time I met Dennis Kibler. I was determined to pursue my interests in parallel logic programming languages; he had contributed to this area. Fortunately, Dennis explained that he wanted to focus on a single research area, something that sounded like "machine earning." He pointed me towards Smith & Medin's book and a way for representing concept descriptions that was more flexible than maintaining "knowledge-compiled" abstractions. This strongly reminded me of my undergraduate thesis advisor's (J. Alan Robinson) interest in intelligent databases (i.e., a large set of data with only a few rules, all encoded as Horn clauses), which allow experts to work more flexibly with domain-specific information than do "knowledge-compiled" expert systems. Something clicked and I found my niche. Dennis is an excellent advisor, always ready with sage advice, able to put issues into perspective, and fair with his students. I greatly appreciate his financial and academic support. Dennis is responsible for originating many of the ideas described in my dissertation; I'm delighted that we collaborated on several projects.

I'd like to thank Mike Pazzani, Rick Granger, and Paul O'Rorke for their support. Mike's experience with issues in cognitive science proved invaluable. I just wish that I had the time to work with him on integrated and relational learning algorithms. Similarly, I regret not having spent time studying brain networks with Rick (maybe in my *next* thesis.) Finally, I learned a great deal from Paul while working with him on abductive EBL algorithms.

The worst thing that happened to me during my stay here was to see Pat Langley take an extended leave. As editor of *Machine Learning*, he was a terrific source of information and spent many hours discussing new advances, research methodology, and unique perspectives on machine learning. Thanks, Pat; you really made a difference.

Three fellow students provided key contributions to this dissertation. First, Marc Albert provided expert help with the mathematical analyses and continually provided encouragement. Similarly, Rob Goldstone became my mentor in cognitive psychology, answering hundreds of questions on the subject. I'm looking forward to continuing our collaboration on the development of psychological models. Finally, Dale McNulty helped develop the GCM-MW model and spent many hours discussing psychological and computational issues on learning and attention.

Every student has their role models. Mine were Kevin Greene, Jeff Schlimmer, and Doug Fisher. Kevin showed me what it took to attain my goals without losing my sanity. Jeff epitomized *class*; everything he touched turned to gold. His work on STAGGER remains an

inspiration. Doug was perhaps my most valuable advocate: his constructive criticism was priceless. Thanks, Doug: I modeled my dissertation after yours.

Kevin was right: an unexpected genius lurks in every university's computer science department. For me there were three: Steve Hampson, Dennis Volper, and Mike Dillencourt, who patiently provided background on connectionist networks, learning theory, and data structures respectively. I'm delighted to have met each of them. Steve was particularly helpful; he spent many hours guiding my understanding on such topics as cognitive psychology and radial basis networks.

My colleagues at UCI provided a stimulating and fun research environment over the years. Those who contributed background for or suggestions on this research include: Stephanie Leif Aha, Kamal Ali, David Benjamin, Jack Beusmans, Tim Cain, Yousri El Fattah, John Gennari, Rogers Hall, David Honig, Wayne Iba, Randy Jones, Jim Kipps, Steve Morris, Bernd Nordhausen, Don Rose, Karen Ruhleder, Stephanie Sage, Wendy Sarrett, David Schulenberg, Karl Schwamb, Glenn Silverstein, Kevin Thompson, Chris Truxaw, Ruediger Wirth, and Jim Wogulis. My officemate David Ruby was especially helpful and was always ready with optimistic encouragement. Thanks to you all and best wishes for a bright future.

Many colleagues at other locations also contributed extensively to my education, including: Ray Bareiss, Gary Bradshaw, Karl Branting, Wray Buntine, Peter Clark, Mark Derthick, Tom Dietterich, Haym Hirsh, Rob Holte, John Kruschke, Joel Martin, Ray Mooney, Andrew Moore, Ross Quinlan, Steven Salzberg, Kingsley Sage, Jude Shavlik, Ming Tan, and Dave Tcheng. Thanks to all – I appreciate your pointers, encouragement, insights, and, yes, even your corrections (Sigh).

The machine learning group at UCI has been blessed by the presence of our administrative assistant *extraordinare* Caroline Ehrlich, also known as the Oregon Cherry Blossom Princess. Her competence, patience, eye for detail, interest, love, and general mothering of UCI's machine learning group are legendary. It's been a joy to have known you, Caroline, and I plan to stay in touch.

Another bright point at UCI is ICS's support group, who responded quickly to my emergencies, provided a reliable computing environment, and have generally been a helpful bunch of folks. Thanks to you all.

Finally, thanks to my family for all their support through the years. If I've made you half as proud of my accomplishment as I am of yours, then I've truly succeeded. I only regret not being as physically close as I feel in spirit: Lala land is a long ways from the east coast. But as long as I'm with Stephanie, I know everything will work out for the best.

This work was supported in part by a gift from the Hughes Aircraft Company.

Curriculum Vitae

Education:

1990: Ph.D. in Computer Science

- Department of Information and Computer Science
- University of California, Irvine
- Dissertation Topic: Instance-Based Learning Algorithms
- Advisor: Professor Dennis F. Kibler

1985: M.S. in Computer Science, Summa Cum Laude
1983: B.S. in Computer Science with Honors, Magna Cum Laude

- Department of Computer and Information Science
- Syracuse University
- Undergraduate Thesis: *Logic Programming Methodology*
- Advisor: Professor J. Alan Robinson

Experience:

1. 1990–91: SERC Visiting Fellow, Turing Institute, Department of Computer Science, University of Strathclyde, Glasgow, Scotland
2. 1986–90: Graduate Student Researcher, Department of Information and Computer Science, University of California, Irvine
3. 1985–87: Teaching Assistant, Department of Information and Computer Science, University of California, Irvine
4. 1984–85: Graduate Student Researcher, Center for Computer Applications and Software Engineering, Syracuse University, Syracuse, NY
5. 1981–83: Teaching Assistant, School of Computer and Information Science, Syracuse University, Syracuse, NY
6. 1982–83: Research Assistant, Logic Programming Research Center, School of Computer and Information Science, Syracuse University, Syracuse, NY
7. Summer 1982: Research Associate, Pattern, Analysis and Recognition Corporation, New Hartford, NY.
8. Summer 1981: Associate Programmer, Fairchild Republic Company, Farmingdale, NY
9. 1981: Programmer, School of Engineering, Syracuse University, Syracuse, NY

Journal Articles:

Aha, D. W. (in press). Tolerating noise, irrelevant attributes, and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies.*

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

Kibler, D., Aha, D. W., & Albert, M. (1989). Instance-based prediction of real-valued attributes. *Computational Intelligence, 5*, 51–57.

Refereed Conference and Workshop Papers:

Aha, D. W., & Goldstone, R. L. (1990). Learning attribute relevance in context in instance-based learning algorithms. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society* (pp. 141–148). Cambridge, MA: Lawrence Erlbaum.

Aha, D. W., & Kibler, D. (1989). Noise-tolerant instance-based learning algorithms. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 794–799). Detroit, MI: Morgan Kaufmann.

Aha, D. W., & McNulty, D. (1989). Learning relative attribute weights for independent, instance-based concept descriptions. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 530–537). Ann Arbor, MI: Lawrence Erlbaum.

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

Kibler, D., & Aha, D. W. (1988). Instance-based prediction of real-valued attributes. In *Proceedings of the Seventh Biennial Canadian Conference on Artificial Intelligence* (pp. 110–116). Edmonton, Alberta: Morgan Kaufmann.

Kibler, D., & Aha, D. W. (1988). Comparing instance-averaging with instance-filtering learning algorithms. In *Proceedings of the Third European Working Session on Learning* (pp. 63–80). Glasgow, Scotland: Pitman.

Weiss, V., & Aha, D. W. (1984). Materials selection with logic programming. In *Proceedings for the 29^{th} Symposium of the Society for Advancement of Material and Process Engineering*, Reno, NA.

Book Chapters:

Kibler, D., & Aha, D. W. (1989). Comparing instance-saving with instance-averaging learning algorithms. In D. P. Benjamin (Ed.) *Change of Representation and Inductive Bias.* Norwell, MA: Kluwer Academic Publishers.

Kibler, D., & Aha, D. W. (1990). Learning representative exemplars of concepts: An initial case study. In J. W. Shavlik & T. G. Dietterich (Eds.), *Readings in machine learning.* San Mateo, CA: Morgan Kaufmann.

Technical Reports and Unpublished Workshop Papers:

Aha, D. W. (1989). *Incremental learning of independent, overlapping, and graded concepts with an instance-based process framework* (Technical Report 89-10). Irvine, CA: University of California, Department of Information and Computer Science.

Kibler, D., & Aha, D. W. (1988). Case-based classification. In *Proceedings of the Case-Based Reasoning Workshop at the Seventh National Conference on Artificial Intelligence* (pp. 62–67). St. Paul, MN: Unpublished manuscript.

Professional Services:

- Organizer, UCI repository of machine learning databases (1987–90)
- Reviewer, *Machine Learning*
- Reviewer, National Science Foundation
- Reviewer, 1990 European Conference on Artificial Intelligence

Departmental and University Services:

- 1988–89: Graduate Student Representative, Faculty Recruitment Committee
- 1987–88: Graduate Student Representative, Computer Resources Committee

Professional Associations:

- Student Member: INNS, AAAI, and ACM

Honors and Fellowships:

- 1989: UCI Dissertation Fellowship
- 1984–85: Texas Instruments Sponsored Fellowship
- 1983: Syracuse University Scholar
- 1983: Who's Who in America's Colleges and Universities
- 1983: Gamma Omicron of Delta Tau Delta Scholarship Award
- 1978: All-American high school cross-country athlete

Abstract of the Dissertation

A Study of Instance-Based Algorithms for Supervised Learning Tasks: Mathematical, Empirical, and Psychological Evaluations

by
David William Aha
Doctor of Philosophy in Information and Computer Science
University of California, Irvine, 1990
Professor Dennis F. Kibler, Chair

This dissertation introduces a framework for specifying instance-based algorithms that can solve supervised learning tasks. These algorithms input a sequence of instances and yield a *partial concept description*, which is represented by a set of stored instances and associated information. This description can be used to predict values for subsequently presented instances. The thesis of this framework is that extensional concept descriptions and lazy generalization strategies can support efficient supervised learning behavior.

The instance-based learning framework consists of three components. The *pre-processor* component transforms an instance into a more palatable form for the *performance component*, which computes the instance's similarity with a set of stored instances and yields a prediction for its target value(s). Therefore, the similarity and prediction functions impose generalizations on the stored instances to inductively derive predictions. The *learning* component assesses the accuracy of these prediction(s) and updates partial concept descriptions to improve their predictive accuracy.

This framework is evaluated in four ways. First, its generality is evaluated by mathematically determining the classes of symbolic concepts and numeric functions that can be closely approximated by *IB1*, a simple algorithm specified by this framework. Second, this framework is empirically evaluated for its ability to specify algorithms that improve IB1's learning efficiency. Significant efficiency improvements are obtained by instance-based algorithms that reduce storage requirements, tolerate noisy data, and learn domain-specific similarity functions respectively. Alternative component definitions for these algorithms are empirically analyzed in a set of five high-level parameter studies. Third, this framework is evaluated for its ability to specify psychologically plausible process models for categorization tasks. Results from subject experiments indicate a positive correlation between a models' ability to utilize attribute correlation information and its ability to explain psychological phenomena. Finally, this framework is evaluated for its ability to explain and relate a dozen prominent instance-based learning systems. The survey shows that this framework requires only slight modifications to fit these highly diverse systems. Relationships with edited nearest neighbor algorithms, case-based reasoners, and artificial neural networks are also described.

Chapter 1
Context, History, and Objectives

Our ultimate objective is to make programs learn from their experience as effectively as humans do. – John McCarthy (1985, page 300)

Most important of all, perhaps, is making such machines learn from their own experience. – Marvin Minsky (1983, page 70)

"What's all this about machine learning?... Who - what madman - would put a computer through twenty years of hard labor to make a cognitive scientist of computer scientist out of it? Let's forget this nonsense - just program it." – Herbert Simon (1983, page 27)

Perhaps the deepest legitimate reason for doing machine learning research is that...learning will turn out to be more efficient than programming... – Herbert Simon (1983, page 36)

Learning is an essential aspect of human behavior. We continually learn to extend and modify our understanding of the environment in which we live. Learning allows us to interact with the world more efficiently over time, whether the task involves reading music, speaking a second language, or controlling our own motor behavior. Learning is generally defined in terms of an improvement in performance for a specified task. For example, Simon's (1983) definition is:

> *Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively the next time.* (page 28)

There are two principal reasons that computer scientists investigate theories and computational simulations of learning: to gain insight on how learning takes place and to implement its benefits in man-made devices such as expert systems and robots. The first reason is the principal objective of *cognitive psychology.* The latter is an objective of the *artificial intelligence* field and is of particular interest to the *machine learning* community. *Cognitive science* attempts to combine these objectives – to create systems that faithfully model aspects of

animate learning behavior. This dissertation describes an investigation of machine learning systems that was inspired by research in cognitive psychology. Most of the algorithms examined here focus on computational concerns and do not attempt to model psychological phenomena. However, one chapter focuses exclusively on modeling human categorization behavior; it should interest cognitive scientists.

The problem investigated in this dissertation concerns the application of instance-based algorithms to incremental, supervised learning tasks. The following sections characterize these tasks and the instance-based approach. Supervised learning is an important subtask for supporting intelligent behavior, especially those involved with forming concepts for the purpose of generating accurate predictions. We continually depend on our ability to make accurate predictions in our daily lives for tasks such as predicting when to change lanes on a freeway, predicting the weight of objects we wish to lift, predicting the time required to complete our tasks, and predicting how others will react to our communications. Since our ability to generate accurate predictions is crucial to determining how well we interact with our environment, methods for learning to improve predictive accuracy are also crucial. Similarly, automated learning methods for improving predictive accuracy are important for robotic and expert systems applications, as well as other tasks of interest to AI practitioners. The instance-based approach has often been applied to supervised learning tasks, but previous research left several questions unanswered, including whether this approach is limited to the applications studied. This dissertation examines the instance-based approach in detail, characterizes the set of supervised learning problems that it can solve, and describes empirical evidence that shows how these algorithms can perform well when confronted with problems such as noisy and irrelevant data. I will also describe results that show how instance-based methods developed for solving supervised learning problems enhance the ability of exemplar-based models of categorization to explain psychological phenomena.

1.1 Taxonomic Categorization of the Approach

Carbonell, Michalski, and Mitchell (1983) presented a useful taxonomy for describing research in machine learning along the following three dimensions:

1. underlying learning strategy,
2. representation of knowledge, and
3. application domain.

1.1.1 Learning Strategy

Several strategies for learning have been studied. These include

- rote learning,
- learning from instruction,
- learning from analogy,
- learning from examples, and
- learning from observation or discovery.

The predominant strategy studied in machine learning has been *supervised learning from examples*, wherein a learner is given a set of training instances, is told that it should predict values for one of the dimensions used to describe these instances, and induces a function called a *concept description* for distinguishing positive and negative examples (Winston, 1975; Mitchell, 1982; Dietterich & Michalski, 1983; Quinlan, 1986a; Clark & Niblett, 1989). That is, they traditionally predict whether instances are members of a given class. The investigations in this dissertation involve an extension of this task; instances will not be restricted to have Boolean-valued *target values* (i.e., "positive" or "negative"). Instead, they can also have numeric- or categorical-valued target values. I will refer to problems involving the prediction of Boolean or categorical target values as *symbolic prediction* tasks. Problems involving the prediction of numeric values will be referred to as *numeric prediction* tasks. In both cases, the source of the instances will be the external environment; the instance generation process will not be controlled either by a teacher or the learning algorithm itself. Nonetheless, I will refer to this as *supervised* learning because the algorithms will be given the target values for the training instances and will be told which values to predict for the test instances. This information is used by instance-based algorithms that save only a subset of the training instances to help determine which instances to retain. Finally, the instances are assumed to be presented sequentially; I will focus on the study of *incremental* rather than non-incremental strategies for supervised learning.

1.1.2 Representation

Many different approaches have been used to represent the functions induced by supervised learning algorithms. Some of these include decision trees (Quinlan, 1986a; Cestnik, Kononenko, & Bratko, 1987; Michie, Muggleton, Riese, & Zubrick, 1984), artificial neural networks (Rumelhart, McClelland, & the PDP Research Group, 1986; Kohonen, 1988; Towell, Shavlik, & Noordewier, 1990), and rules (Michalski, Mozetic, Hong, & Lavrač, 1986; Clark & Niblett, 1989; Holland, 1986; Wilson, 1987). The representation studied in this

dissertation is one called the *instance-based* approach, wherein target value predictions are derived solely from specific instance information.[1]

1.1.3 Application Domain

The algorithms studied in this dissertation are *not* domain-specific. That is, they were not constructed to work for a pre-determined application or set of applications. However, they can be tuned to exploit domains-specific knowledge. Chapter 7 surveys instance-based algorithms of this genre. Therefore, I will not refer to the algorithms described in this dissertation as *empirical* learning algorithms, lest the framework introduced in Chapter 2 be incorrectly identified as specifying only knowledge-poor algorithms.

1.2 The Instance-Based Approach in Perspective

This dissertation investigates an alternative learning strategy called *instance-based learning* (IBL), which is a form of *exemplar-based learning* where abstractions are not maintained for the purpose of solving prediction tasks. These algorithms can be interpreted as *case-based reasoning* algorithms in that they rely on specific instances. However, IBL algorithms focus primarily on learning issues, which are only one of many concerns addressed by case-based reasoning algorithms. Nonetheless, several IBL algorithms address additional issues that are also of interest to the case-based reasoning community. These relationships will be discussed further in Section 7.3.1.

1.2.1 Characterization of Instance-Based Learning

The IBL approach differs from other methods for solving supervised learning tasks in several respects.

- IBL algorithms have an *extensional* representation; concepts are represented by their exemplars and are not assumed to be conjunctive.
- IBL algorithms are *lazy generalizers.* In comparison with other approaches, they do less work at the time a training instance is presented and more work when predictions are required. Thus, they are appropriate for applications where the probability is low that generalizations derived from training instances at the time of their presentation will be used to derive predictions for subsequently presented instances.

[1]This is not quite true. Effective instance-based learning algorithms associate additional information with stored instances or the predictor attributes describing them. This is discussed further in Chapter 2.

- IBL algorithms have a *general learning bias*. Unlike some learning algorithms, they do not require that symbolic concepts be hyper-rectangular in shape. Instead, they assume that concepts have hyper-polygonal shapes. Thus, they will tend to learn more quickly than rule-based or decision tree learning algorithms when the target concept is not easily describable by hyper-rectangles.
- IBL algorithms naturally support efficient methods for incremental learning (Schlimmer & Fisher, 1986; Utgoff, 1989). Updating concept descriptions consists mainly of adding, deleting, or editing stored instances.
- IBL algorithms naturally support partial matching; they calculate the *similarity* between pairs of instances. Thus, these algorithms can represent *graded concepts*, where concept members differ in their *typicality ratings*. An instance's rating is defined as an increasing function of its similarity to instances in the same concept and a decreasing function of its similarity to all other instances.

All of these distinguishing characteristics are also true, to some degree, for the more general class of exemplar-based learning algorithms.

1.2.2 Historical Overview in Machine Learning

A simple IBL algorithm is the *k-nearest neighbor* (k-NN) pattern classifier (Fix & Hodges, 1951), which classifies an instance according to the majority classification of its k nearest neighbors. Several AI researchers have included k-NN in their comparison studies of learning algorithms because it is well-known, easy to implement, and serves as a good *strawman* algorithm. For example, Shepard (1983) tested several algorithms on a task involving the classification of chocolate images. He was satisfied that the ACLS decision tree algorithm could achieve accuracies on par with k-NN, which he called a "practical alternative classifier." Shepard noted that k-NN had a faster learning rate but could not yield an intelligible summary of the concept being learned. Weiss and Kapouleas (1989) included the 1-NN algorithm in their large set of comparison algorithms. They found that it performed well when given relevant attributes but poorly when instances are described by irrelevant attributes. Towell, Shavlik, and Noordewier (1990) included 1-NN in their comparison of five learning algorithms to a problem in molecular biology. They found that, although it outperformed ID3 (Quinlan, 1986a), it performed more poorly than both backpropagation (Rumelhart *et al.*, 1986) and an algorithm that was given domain-specific information. Finally, Robinson (1989) compared the 1-NN algorithm and 15 different connectionist algorithms on a task involving vowel classification. Surprisingly, he found that 1-NN performed best and suggested that this result poses a challenge to improve the abilities of connectionist algorithms. This does not always occur; Robinson also found that 1-NN did not outperform all connectionist algorithms in other applications. However, none of these researchers explored extensions of k-nearest neighbor algorithms that could improve their learning behavior. This is understandable since they were not proponents of the instance-based approach.

The first proponent of the instance-based approach in the machine learning literature was Bradshaw (1985, 1986, 1987), who introduced the NEXUS speech recognition system. He showed that the IBL approach could be used for practical applications and detailed an instance-averaging method that reduces storage requirements. There have since been five other prolific proponents of the exemplar-based approach for supervised learning tasks. First, Bareiss and his colleagues developed Protos and applied it to a clinical audiology task (Porter & Bareiss, 1986; Bareiss, Porter, & Wier, 1987; Bareiss, 1989a; 1989b). It was the first learning apprentice (Mitchell, Mahadevan, & Steinberg, 1985) that used the exemplar-based approach and its contributions included data structures and procedures for indexing domain specific information so that it can be retrieved efficiently and effectively. Second, Stanfill developed the MBRtalk word pronunciation system with Waltz (Stanfill & Waltz, 1986; 1988; Stanfill, 1987; 1988; Waltz, 1990). Their central contributions include highlighting the parallel nature of IBL algorithms and introducing a function that can compute similarities for symbolic attribute values. Third, Clark (1988) published the only paper comparing exemplar-based and rule-based learning algorithms in the machine learning literature. He subsequently developed Optimist (Clark, 1989), an exemplar-based system for appraising oil prospecting sites that is currently being used by the Enterprise Oil Company. Fourth, Salzberg (1988; 1990) developed NGE, which derives hyper-rectangular abstractions from instances, and also helped develop PEBLS (Cost & Salzberg, 1990). These algorithms recorded good classification accuracies on several medical diagnosis applications. Finally, I have worked with Kibler and Albert on the development of a comprehensive sequence of IBL algorithms; we have evaluated them mathematically and empirically on several symbolic and numeric prediction tasks (Kibler & Aha, 1987; 1988; 1989; Aha & Kibler, 1989; Kibler, Aha, & Albert, 1989; Aha, Kibler, & Albert, 1991; Aha, 1989a; in press).

Exemplar-based learning algorithms have been successfully applied to several challenging supervised learning applications, including such problems as speech recognition, word pronunciation, power load prediction, oil prospecting appraisal, medical diagnosis, learning to control robotic movements, and handwritten character recognition. However, most of the research on these learning algorithms in the machine learning literature have been limited to case-study demonstrations. Moreover, there have been no attempts to survey the field in an attempt to relate the myriad of IBL methods that have been used to solve these many problems.

As a first attempt, these exemplar-based algorithms can be related and differentiated by whether they generalize their training instances, whether they store all training instances, and whether they encode a sizable amount of domain-specific knowledge. Figure 1.1 displays a simple decision tree that uses these dimensions to relate several of the more popular exemplar-based learning algorithms that have been developed since 1985. However, the decision tree in Figure 1.1 does not completely succeed in relating these algorithms; several of them would be grouped differently if a different ordering on the tests or a different set of attribute tests had been used. This tree hides several important relationships between these systems. For example, both Protos (Bareiss, 1989a) and Optimist (Clark, 1989) are

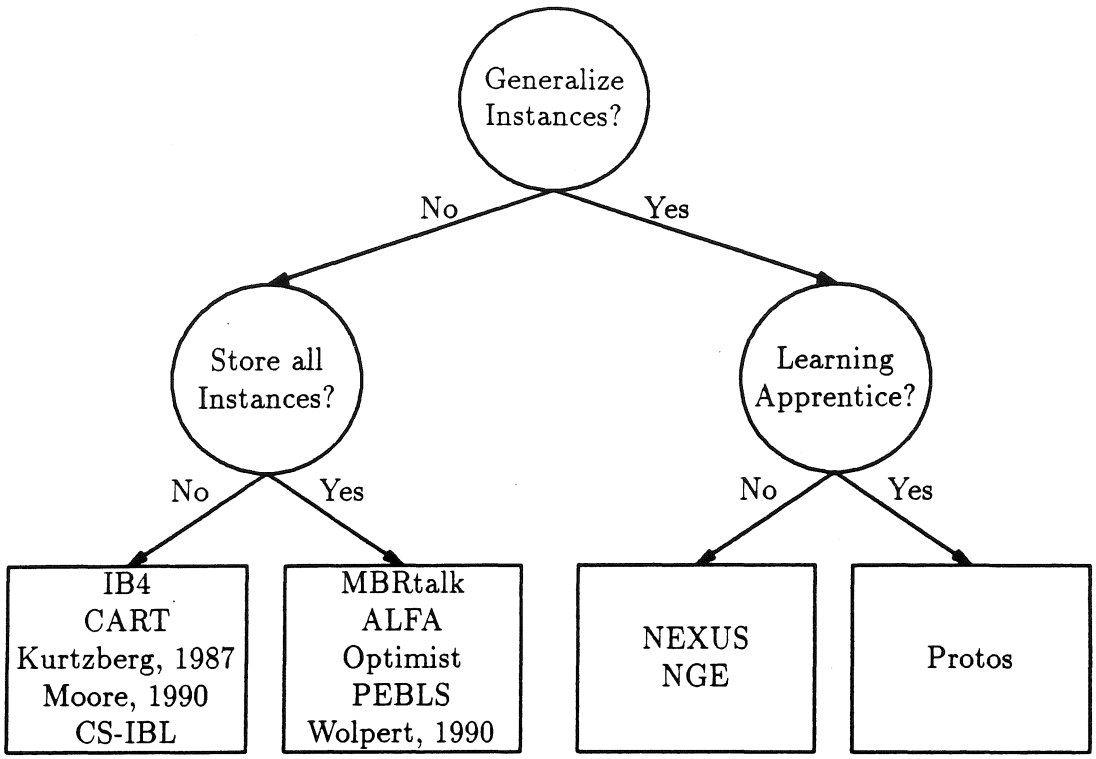


Figure 1.1: Relationships among exemplar-based learning systems.

taught information on how to compute the similarity between two instances, most of these algorithms attach weights with attribute dimensions describing the instances, and several of these algorithms can predict numeric values. Table 1.1 lists the values of nine properties for six of these algorithms. All of the algorithms mentioned in Figure 1.1 and Table 1.1 will be surveyed in Chapter 7 except IB4 (Aha, 1989a), which will be evaluated in Chapters 4 and 5. I will introduce a more formal framework for relating these algorithms in Chapter 2.

1.2.3 Relationship to Other Fields

The instance-based approach has close ties with ongoing research in both pattern recognition and psychology. In fact, this dissertation was directly inspired by research in both fields.

Researchers in the pattern recognition community have long understood the utility of making predictions based on a stored set of similar instances. IBL algorithms are derived from variants of k-NN algorithms (Fix & Hodges, 1951; Sebestyen, 1962; Cover & Hart, 1967; Cover, 1968; Duda & Hart, 1973). These algorithms represent an instance with a set

Table 1.1: An abbreviated instance-based description of six exemplar-based learning systems.

Property	Optimist	Protos	NGE	IB4	MBRtalk	NEXUS
Predicts Symbolic Values		√	√	√	√	√
Predicts Numeric Values	√			√		
Uses Attribute Weights	√	√	√	√	√	
Generalizes Instances		√	√			√
Stores All Instances	√				√	
Taught Similarity Fn.	√	√				
Learns Similarity Fn.		√	√	√		
Interactive	√	√				
Cognitively Plausible						

of n attribute-value pairs, which can be interpreted as a vector or point in an n-dimensional space. They predict an instance's target value from the target values of its k nearest neighbors in this space. The similarity of two instances can be measured in many ways, although it is commonly defined as the inverse of their Euclidean distance. Cover and Hart proved that the 1-nearest neighbor algorithm's expected misclassification rate does not exceed twice the Bayes optimal rate, which is a lower bound on misclassification rates given complete probability density information for every point in the space. They also showed that the difference between the misclassification rates of the k-nearest neighbor algorithm and the Bayes rate decreases exponentially with increasing values of k. Several researchers have subsequently investigated the behavior of *edited k-nearest neighbor algorithms* (Hart, 1968; Gates, 1972; Wilson, 1972; Penrod & Wagner, 1977; Dasarathy, 1980; Devijver, 1986; Voisin & Devijver, 1987). The emphasis of their investigations was to show that these algorithms can decrease the time to convergence (i.e., the Bayes optimal misclassification rate). Although many mathematical and theoretical successes have been reported, the goals of pattern recognition researchers differ from those of the machine learning community. For example, a main concern with edited nearest neighbor algorithms is that they perfectly classify the training set; few researchers who have studied edited k-NN algorithms have addressed issues of noise, irrelevant attributes, and overfitting effects (Niblett & Bratko, 1986). Also, most of these algorithms repeatedly process instances and cannot process symbolic attribute values. In summary, the IBL approach has been studied in the pattern recognition literature, but with different goals and perspectives. This body of research is surveyed in Section 7.1.

The IBL approach has also been studied in earnest by the cognitive psychology community, who were originally inspired by Sebestyen's (1962) work on the nearest neighbor pattern classification algorithm (Reed, 1970). The goals of this community include building models to explain observed psychological phenomenon associated with the processes of categorization. Three views of concepts have dominated the design of these models (Smith & Medin, 1981). First, the *classical view* assumes that concepts are conjunctive and that instances do not vary in their typicality. As an example, Mitchell's (1982) work on version

spaces assumed this viewpoint. Second, the *probabilistic view* also represents concepts with a centralized description, but attributes are assumed to have a probabilistic relationship to concepts. This view is well-represented in the machine learning literature (Rumelhart *et al.*, 1986; Schlimmer, 1987a; Fisher, 1987; Gennari, 1990; Langley, Thompson, Iba, Gennari, & Allen, 1989). The final view is the *exemplar-based view*, which has been under-represented in the machine learning literature. Although preliminary evaluations of exemplar-base models of categorization suggested that they provided poor fits to subject data, research beginning with Medin and Schaffer's (1978) investigations of more elaborate exemplar-based models have suggested otherwise (Medin & Schwanenflugel, 1981; Medin, Altom, Edelson, & Freko, 1982; Medin, Dewey, & Murphy, 1983; Hintzman & Ludlam, 1984; Busemeyer, Dewey, & Medin, 1984; Nosofsky, 1984; 1986; 1987; 1989; Hintzman, 1984; 1986; 1988; Medin & Edelson, 1988; Nosofsky, Clark, & Shin, 1989). In fact, exemplar-based models have explained more psychological phenomenon than any other model of categorization. However, researchers in this field rarely attempt to design process components (i.e., learning algorithms) for their models. Therefore, few of these models have been evaluated in terms of their computational efficiency. This body of research is surveyed in Section 6.3.

1.3 Objectives of this Dissertation

As portrayed in Figure 1.2, IBL algorithms have ties with pattern recognition, cognitive psychology, and machine learning. The primary objectives of this dissertation contribute to each of these areas, although its emphasis is on machine learning, cognitive psychology, and pattern recognition in decreasing order:

1. *Paradigm Framework:* The relationships of IBL algorithms have rarely been systematically reported. Since their number continues to multiply, it is important to have some means for relating them and for describing improvements that are of general interest to both practitioners and researchers. Therefore, I will describe a framework for instance-based learning algorithms[2] that presents a coherent view of the IBL prediction process. This framework will be used to guide the development of IBL algorithms: improvements will be described by how they improve on a specific component function of the framework. The IBL framework will also prove useful as a means for relating the multitude of existing IBL algorithms in the machine learning literature.
2. *Mathematical Analyses of Generality:* No effort has been made by other groups to determine what IBL algorithms can learn efficiently. Therefore, I will present theorems that describe the classes of symbolic concepts and numeric functions that a simple IBL algorithm can probably, approximately learn with high confidence (i.e., with high probability).

[2]This framework applies to exemplar-based learning in general, but I wanted to avoid using the acronym *EBL*, which commonly refers to *explanation-based learning* (Mitchell, Keller, & Kedar-Cabelli, 1986).

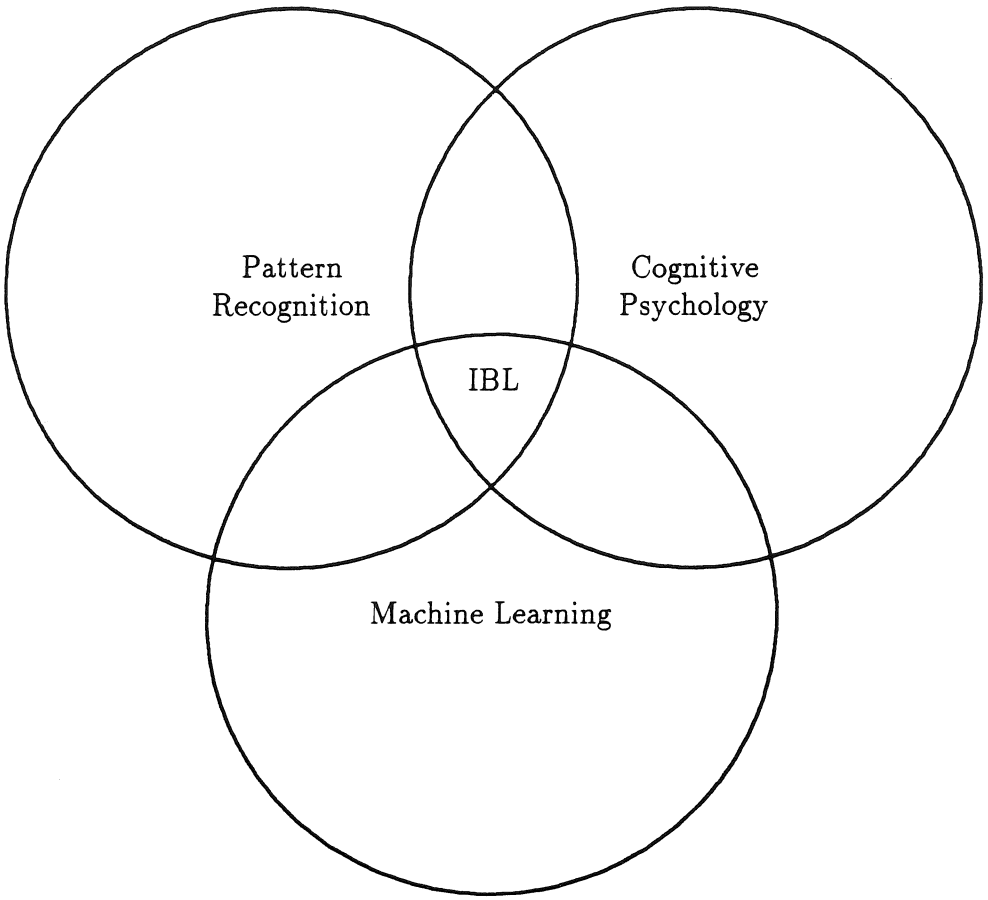


Figure 1.2: IBL algorithms have been studied in at least three areas.

3. *Empirical Analyses of Efficiency:* Most empirical investigations of IBL algorithms have been case studies with few attempts to understand the limits of the algorithms' behavior, especially their sensitivity to challenging domain characteristics.[3] Therefore, I will describe empirical investigations for IBL algorithms that address the issues of storage requirements, noise, and irrelevant attributes respectively. Studies with artificial domains will be described that demonstrate the behavior of the algorithms under carefully controlled conditions. Studies with several database applications will be described to demonstrate that these algorithms solve their respective problems in more practical settings. I will also conduct detailed investigations of these algorithms' high

[3]Exceptions include Stanfill's (1987) study of MBRtalk's ability to tolerate noise and a few irrelevant attributes for the word pronunciation task, Bareiss's (1989b) lesion study investigations of Protos for the clinical audiology application, and Tan and Schlimmer's (1990) investigations of their algorithms' behavior with respect to irrelevant attributes and varying attribute evaluation costs.

level parameters to determine the range of settings for which good performance can be obtained.

4. *Psychological Plausibility Analyses:* IBL algorithms have been developed in the machine learning literature without feedback or concern with psychologically plausible models of categorization. Therefore, I will describe simulations of and experiments with four exemplar-based process models whose designs are constrained by work in *both* cognitive psychology and machine learning. The objective of the simulations is to show that results from machine learning research on IBL algorithms can contribute to the literature on psychologically plausible models. The objective of the experiments is to analyze these models comparative abilities to fit subject data. These results will show that significantly better fits can be obtained by attending differentially to attributes in a context-dependent manner. This is a new contribution to research on psychologically plausible exemplar-based models: this behavior has not previously been displayed by a fully-specified exemplar-based process model.

These objectives all concern a form of evaluation: evaluation of a framework for its ability to clarify design decisions, a mathematical evaluation of a simple IBL algorithm's generality, an empirical evaluation of a set of IBL algorithms' ability to solve challenging problems in supervised learning, and a comparative evaluation of an IBL algorithm's ability to fit data collected from human subjects.

1.4 Methodology

Several frameworks have been proposed for relating different methodological perspectives in artificial intelligence research (e.g., Newell, 1973; Feigenbaum, 1977; Ringle, 1979). More recently, Hall and Kibler (1985) developed a detailed topology after careful consideration of six previous frameworks. They used four dimensions for contrasting different methodological perspectives:

1. *Artificiality*: This distinguishes research that attends to psychologically plausible constraints from research that develops "intelligent" systems without these concerns.
2. *Generality*: This distinguishes research targeted for specific applications or cognitive functions from research that attempts to model general reasoning mechanisms or general principles of cognitive behavior.
3. *Methodological direction*: This distinguishes research that is characterized by a bottom-up construction of systems from a more formal top-down decomposition. Hall and Kibler consider the work on DENDRAL (Buchanan, Sutherland, & Feigenbaum, 1970) to be an example of the bottom-up direction and Hart, Nilsson, and Raphael's (1968) work on search in heuristic graphs to be an example of top-down construction.

4. *Method of verification*: This distinguishes research that focuses on demonstrating the correspondence between computational simulations and collected subject data from research that is cognitively motivated but lacks these demonstrations.

Hall and Kibler used their framework to distinguish five different classes of research perspectives:

1. *Performance*: artificial, specific applications
2. *Constructive*: artificial, general principles, bottom-up approach
3. *Formal*: artificial, general principles, top-down approach
4. *Speculative*: natural, no detailed demonstrations of correspondence
5. *Empirical*: natural, detailed demonstrations of correspondence

Example of these approaches are R1 (McDermott, 1982), ARCH (Winston, 1975), A* (Hart, Nilsson, & Raphael, 1968), SAM (Schank & Abelson, 1977), and ACT* (Anderson, 1983) respectively. Fisher (1987) described a modification of this topology that demotes the importance of the artificiality dimension and emphasizes an information-processing viewpoint.

Hall and Kibler noted that researchers can shift their perspectives over time. Moreover, research cannot necessarily be categorized into only one of these perspectives; Fisher (1987) and Mooney (1987) both noted that their respective methodologies adopt several of these five perspectives. Likewise, the research presented in this dissertation also exhibits multiple perspectives:

1. *Constructive*: The sequence of four comprehensive algorithms described in Chapter 4 were developed in a bottom-up fashion with the goal of improving the robustness of their behavior. Issues of cognitive plausibility were ignored.
2. *Formal*: A general framework will be introduced for unifying a body of disparate work (i.e., exemplar-based learning algorithms). Formal proofs of convergence for a simple algorithm specified by this framework will also be described.
3. *Empirical*: IBL algorithms constrained by known psychological phenomena will be evaluated for their fits to subject data.

The contributions of this dissertation cannot be viewed from a performance perspective since no single application provides a focus for the development of the framework or the algorithms that will be evaluated. Likewise, these contributions cannot be viewed as using a speculative methodology; psychological models will be evaluated according to their ability to simulate data collected from human subjects.

Hall and Kibler's (1985) topology is described by a decision tree. The four dimensions on which to judge perspectives are assumed to be Boolean in nature (i.e., natural or artificial). However, some of these dimensions would be better interpreted as having continuous values

(e.g., the artificiality dimension); this is an example of a *numeric discretization process* that is often used with decision trees. Also, their tree tends to hide relationships, such as a constructive perspective for research inspired by cognitive science. The four cognitive models described in Chapter 6 are an example of this perspective; I will define the models of categorization using a bottom-up approach. Decision trees, and abstractions in general, are useful for providing summary descriptions. However, they are not necessarily robust or useful when they are asked to provide detailed information characterizing specific instances. This dissertation's theme argues that the storage of specific instances can help to support these capabilities. Therefore, perhaps a better way to characterize my research perspectives is to use the simple instance-based strategy proposed by Carbonell *et al.* (1983). They argued that research in machine learning is organized around three foci:

1. task-oriented studies,
2. cognitive simulation, and
3. theoretical analysis.

All three of these methods are used to evaluate IBL algorithms in this dissertation.

1.5 Overview of this Dissertation

This dissertation describes a unifying framework for and evaluations of instance-based learning algorithms. Chapter 2 introduces the IBL framework and IB1, a simple algorithm that the framework specifies. Detailed examples are provided to explain how IB1 incrementally improves its predictive accuracy for both symbolic and numeric prediction tasks. The performance dimensions used to evaluate IBL algorithms in the rest of this dissertation are also presented.

Chapter 3 describes convergence proofs for IB1 for both symbolic and numeric prediction tasks. They are distribution free, apply to any value of k (i.e., the number of nearest neighbors used to derive predictions), and apply to any number of attribute dimensions. The proofs show that IB1 can PAC-learn the class of concepts that are describable as a finite union of finite-sized regions in the instance space and the class of continuous functions with bounded slope. Efficiency concerns are also addressed.

Chapter 4 describes empirical investigations with three extensions of IB1, named IB2, IB3, and IB4. These three algorithms will address the tasks of reducing storage requirements, tolerating noise, and learning relative attribute relevance respectively. Their relationships will be detailed using the IBL framework.

Chapter 5 describes five parametric studies for these IBL algorithms. The parameters chosen correspond to important functions and parameters of the IBL framework, including

the function for processing symbolic attributes, the function for processing missing values, and the value for k. Each investigation includes both a lesion study and an examination of at least one intuitively pleasing alternative definition.

Chapter 6 surveys the literature on psychological models of categorization, introduces four psychologically plausible exemplar-based process models, and evaluates them in comparison simulations and on their ability to fit data collected from two experiments with human subjects. Both the survey and the experimental results suggest a positive correlation between the ability of a model of categorization to utilize attribute correlation information and the number of psychological phenomena it can model.

Chapter 7 describes a survey of related research on pattern recognition algorithms, IBL algorithms in the machine learning literature, and related learning paradigms that specify algorithms which exploit specific instance information. The IBL framework is used to relate the IBL algorithms surveyed.

Chapter 8 finishes with a summary of contributions, a discussion of the limitations and advantages of the framework, and suggestions for future research goals.

Chapter 2
Framework

This chapter introduces instance-based learning (IBL) algorithms. The focal point of this introduction is the framework for this class of algorithms, which is detailed in Section 2.2. A simple IBL algorithm, named *IB1*, is described in Section 2.3 to exemplify how the components of this framework can be instantiated. Section 2.4 summarizes how the performance of IBL algorithms is evaluated in Chapters 4–6. Section 2.1 begins by defining the terminology used in this dissertation to describe how IBL algorithms operate.

2.1 Terminology and Representation

The topic of this dissertation is the study of instance-based learning algorithms as applied to incremental, supervised learning tasks when the source of the instances is the external environment. That is, the algorithms studied here passively accept instances from an exterior process rather than request or choose specific ones for input. Each instance is assumed to be represented by a set of attribute-value pairs. Many supervised learning algorithms have employed this representation to describe instances (e.g., Mitchell, 1982; Breiman, Friedman, Olshen, & Stone, 1984; Michalski *et al.*, 1986; Quinlan, 1986a; Sejnowski & Rosenberg, 1987; Wilson, 1987; Bradshaw, 1987). This is also the common representation for instances in the literature on computational learning theory (e.g., Valiant, 1984; Kearns, Li, Pitt, & Valiant, 1987; Haussler, 1987). Most studies of concept formation in the literature on experimental psychology also adopt this representation for instances (Hayes-Roth & Hayes-Roth, 1977; Medin & Schaffer, 1978; Gluck, Bower, & Hee, 1989). The attribute-value representation is appealing because it is simple and amenable to analyses. However, it is restrictive: attribute-value pairs do not encode relationships among sets of attributes that can be used to assist in making accurate predictions. Branting (1989), among others, has consequently advocated

using semantic networks to represent instances.[1] Although these representations would be useful, they have not yet been used to represent instances for IBL algorithms.

Instances that are described by n attributes are points in an n-dimensional *instance space*. Each attribute is defined over a set of totally-ordered (e.g., numeric) or unordered (e.g., symbolic) values. Functions for processing elaborate types of values, including internal disjunction (Michalski, 1983) and structured attribute values (Wasserman, 1985; Stepp & Michalski, 1986; Thompson & Langley, 1989), have not yet been implemented. However, IBL algorithms can tolerate missing attribute values (see Section 2.3).

Traditionally, exactly one of the attributes used to describe instances is a distinguished *target attribute*. The remaining attributes are *predictors*. An important purpose of the learning algorithm is to learn to accurately predict the target attribute's value when given an instance whose target value is missing. As mentioned earlier, IBL algorithms can learn multiple concepts simultaneously. Therefore, they are told which *set* of attributes are targets and which set of attributes are predictors (for each target). Target attributes are traditionally assumed to have only two possible values (e.g., "positive" and "negative"). IBL algorithms instead allow them to have either totally-ordered or unordered values.

The set of all instances in the instance space that have the same value for a target attribute form a *category* (or *class*). A *concept* is a purposeful description of a category (Matheus, 1987).[2] IBL algorithms are given a sequence of *training instances* and learn concepts for each symbolic-valued target attribute for the purpose of accurately predicting the targets' values. Concepts are functions that map instances to categories. Given an instance x drawn from an instance space, a concept learned for target attribute t yields a prediction for x_t, the value that x is predicted to have for attribute t.

IBL algorithms learn a different type of mapping for numeric-valued target attributes. In these situations, there can be a large number of distinct numeric target values in the data. The notion of a concept is not as useful here because it doesn't capture or exploit the total ordering on the numeric-valued target attribute. Therefore, IBL algorithms make explicit use of this information when learning to predict numeric values. However, it is still convenient to refer to these learned numeric functions as concept descriptions. Section 2.3 includes detailed examples that shows how an IBL algorithm learns accurate concept descriptions for both symbolic- and numeric-valued targets.

[1] I followed Bradshaw's (1987) lead here in referring to input objects as *instances* rather than *examples*. Another reason to refer to them as instances is to avoid confusion with the term *exemplar*, which Smith and Medin (1981) used to refer to both specific instances and sets of specific instances, otherwise known as *exemplars*. The algorithms that will be investigated in this dissertation input specific instances rather than exemplars. Also, I avoid the term *case* here to avoid confusion with its meaning in the case-based reasoning literature (Hammond, 1989), where Branting and others have argued that attribute-value representations are inadequate for challenging analogical problem-solving tasks.

[2] I will use the terms "concept" and "concept description" synonymously as a matter of convenience.

2.2 The IBL Framework

My goal in this dissertation is to introduce, evaluate, and explicate the utility of the instance-based paradigm for supervised learning tasks. This dissertation does *not* focus on a particular learning algorithm, but rather focuses on a computational *paradigm* from which an infinite number of learning algorithms can be specified. A sound way to describe a paradigm for learning is to present a concise yet illuminating framework for it, which is the topic of this section.

A framework serves several purposes. First, it furnishes context so that the reader can gain insight to the perspective from which the framework evolved. Second, it provides an organizational structure that explicates the differences between algorithms specified by the framework. Third, frameworks can also be used to compare algorithms specified by different computational paradigms. They also help to distinguish the relative capabilities and limitations of these paradigms. Finally, frameworks are valuable because they serve as a tangible foundation that can be used by other researchers to improve the paradigm.

Learning involves some improvement in the performance of some task (Langley, 1987; Kibler & Langley, 1988). Therefore, it is useful to partition the framework for IBL algorithms into two sets of components: performance and learning. Both components are used during training but only the performance component is used during testing. Both components use attribute information (i.e., attribute typing information and predictor/target labeling) to control their respective processes. The performance task, in this case, is to yield predictions for target values given a sequence of training instances. The learning task is to develop a concept for each of the target attributes. Figure 2.1 displays this abstract framework for the case where there is only one target concept. Figure 2.2 details the performance component. These figures use an SADT-style diagram to describe the framework pictorially (Ross, 1977). The boxes represent functions while the arrows refer to data structures that are passed among them. Arrows on the left, right, and top of a box are input, output, and control information respectively. This diagram shows that the performance component inputs a pre-processed training instance and a *partial concept description*, which includes a set of stored instances and, in more elaborate IBL algorithms (e.g., IB3 in Section 4.3 and IB4 in Section 4.4), prediction-related information such as attribute weight settings and a record of each stored instance's prediction accuracy. The performance component outputs a prediction for the target attribute's value. The learning component inputs the partial concept description, the processed training instance, and the prediction. It outputs a modification of the partial concept description.

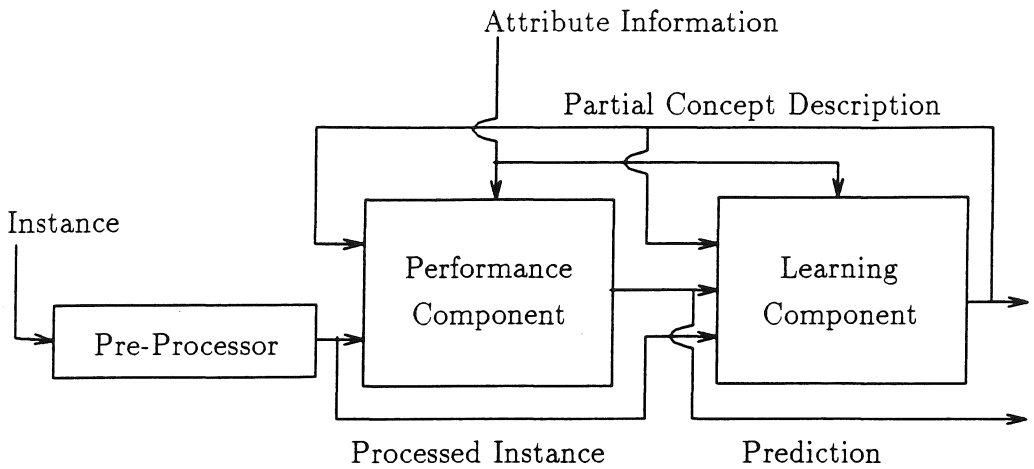


Figure 2.1: The IBL framework.

The following four subsections briefly describe the four basic functions that constitute the IBL framework:

1. *Normalizer*: Normalizes the numeric-valued predictor attributes
2. *Similarity*: Computes the similarity of two instances
3. *Prediction*: Generate a prediction, and
4. *Memory updater*: Updates the partial concept description

Each of these functions is used by one of the three IBL framework components. Normalization is the responsibility of the pre-processor component. The performance component is responsible for computing similarities and generating predictions. Finally, the memory updating function is the primary function of the learning component.

2.2.1 Normalization Function

The pre-processor component maintains and updates summary information concerning the values of each numeric attribute during the training process. This information is used by the *normalizer function*, which normalizes each numeric attribute's values. This forces the similarity function to assume that each of these attributes has the same range of values. This normalization process creates a bias; each attribute is assumed to have the same relevance for the purpose of the performance and learning components. In Section 4.4, I will argue that this is rarely true for most databases. However, this is a fair decision without additional

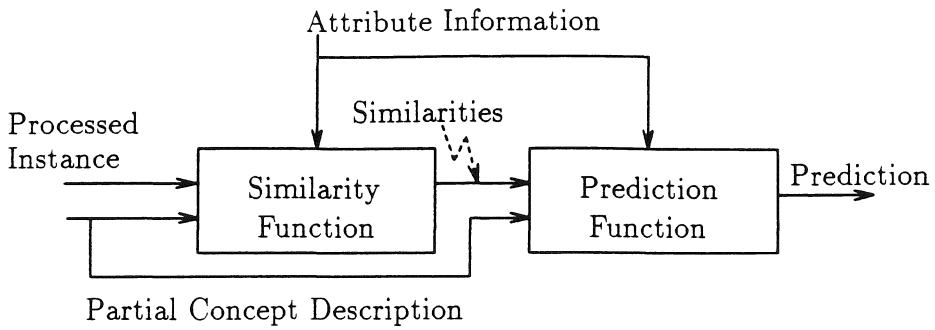


Figure 2.2: Performance component for the IBL framework.

information concerning the relative relevance of attributes for the purpose of the learning algorithm.

Re-normalization is required whenever determined by the pre-processor as new information is collected from subsequently processed instances. A simple example of a normalization function is one that maintains the highest and lowest values for each numeric-valued attribute and uses them to linearly normalize these attributes.

2.2.2 Similarity Function

IBL algorithms are *not* limited to rote learning, defined as memorization without capability for inductive generalization (Carbonell *et al.*, 1983). Instead, IBL algorithms derive predictions from a partial concept description using the two functions in the performance component. Its first function is the *similarity function*, which acts on a partial concept description and a pre-processed instance. It calculates and outputs the numeric-valued *similarity* of the novel instance with each instance in the partial concept description. A simple similarity function is one that computes the inverse of the Euclidean distance of two instances x and y in the instance space, where the "distance" between two values of a symbolic-valued attribute is defined to be 0 if they are identical and 1 otherwise:

$$\text{Similarity}(x, y) = \frac{1}{\sqrt{\sum_{i \in P} \text{Attribute\_difference}(x_i, y_i)}}, \tag{2.1}$$

where

$$\text{Attribute\_difference}(x_i, y_i) = \begin{cases} (x_i - y_i)^2 & i \text{ is numeric-valued} \\ x_i \neq y_i & \text{otherwise} \end{cases} \tag{2.2}$$

and where the instances are described by the set P of predictor attributes.

2.2.3 Prediction Function

The performance component's second function is the *prediction function.* This inputs the computed similarities and outputs a prediction of the target attribute's value for the instance given to the performance component. A simple prediction function, which can be used to predict either symbolic or numeric values, is the *nearest neighbor prediction function*, which outputs the target attribute value of the instance in the partial concept description with the highest similarity.[3]

2.2.4 Memory Updating Function

The primary function in the learning component is the memory updater, which updates the partial concept description after each training instance is processed. As mentioned earlier, the partial concept description consists of a set of instances and, possibly, additional information related to the utility of the stored instances and the attributes that describe them. I do not mean to preclude the possibility that other types of useful information may be maintained by this function (e.g., parameters for the prediction function). However, I do mean to constrain the functionality of the memory updater to update either the partial concept description, the similarity function, or the prediction function.

The most elementary memory updating function simply stores all training instances into the partial concept description.

2.2.5 Summary of the Framework

The performance and learning components in this framework are tightly coupled. The performance component yields predictions that determine how the learning component is updated. The learning component updates the partial concept description used by the performance component. In some IBL algorithms, the partial concept description even contains parameters for the functions in the performance component. Thus the outputs of one component strongly influence the processing of the other.

This framework is relatively general. Several types of normalization, similarity, prediction, and learning functions can be used to instantiate an IBL algorithm. Chapter 4 describes three IBL algorithms that are variants of IB1, which is detailed in the next section, that differ in terms of their selection of these three latter functions. Chapter 5 investigates several additional IBL algorithms in which the component definitions are varied to determine

[3] If at least two instances tie in maximal similarity, then a random choice is made to select one of them to be used to generate a prediction.

Table 2.1: The IB1 training algorithm.

```
Key: T: Training set
     P: Set of predictor attributes
     PCD: Partial concept description

Train(T, P)
     1. Set global variables
     2. for each x_i ∈ T do
        2.1 x_i ← Pre_process(x_i, P)
        2.2 Learn(x_i)

Set global variables:
     1. PCD ← ∅
     2. Initialize min and max values for each predictor attribute

Pre_process(x, P)
     1. if (x is a training instance)
        then Update_bounds(x)
     2. return Normalize(x)

Learn(x)
     PCD ← PCD ∪ {x}
```

whether the originally selected definitions were the best choices. Finally, Section 7.2 uses this framework to relate a dozen exemplar-based algorithms that have been described in the machine learning literature. In summary, the IBL framework specifies a large set of possible IBL algorithms. However, this framework has several limitations and restrictions. For example, some useful extensions include using an indexing mechanism to minimize the number of instance and attribute references required to compute similarities. Chapter 8 examines the capabilities and limitations of this framework more extensively.

2.3 The IB1 Algorithm

This section introduces the *IB1* (Instance-**B**ased **1**) algorithm, whose training algorithm is detailed in Table 2.1. IB1 is a simple instance-based learning algorithm that closely resembles the k-nearest neighbor decision rule (Fix & Hodges, 1951; Cover & Hart, 1967; Duda & Hart, 1973). However, they are not identical; IB1 normalizes its instances and, as described in the next paragraph, IB1 has a policy for tolerating missing attribute values.

2.3.1 IB1's Component Functions

IB1's training algorithm inputs a training set of instances and the set of predictor attributes. IB1 begins by initializing some global variables: it sets its partial concept description to be empty, sets the "observed" lower bound for each numeric-valued predictor attribute to infinity, and sets the "observed" upper bound for each numeric-valued predictor attribute to negative infinity. Training on an instance involves only pre-processing and learning. Predictions aren't generated during training.

Pre-processing: IB1's pre-processor keeps a record of the highest and lowest values known for each numeric-valued attribute. Whenever the range of observed values for a numeric-valued attribute a is extended by the value in a newly observed training instance, its value in all the stored instances is re-normalized. IB1 employs a linear normalizing function. Attribute a's unnormalized values in instance x is normalized using Normalize(x). This calls subfunction Normalize_attribute for each numeric predictor attribute a and normalizes its value using $x_a \leftarrow$ Normalize_attribute(x_a, a), where:

$$\text{Normalize\_attribute}(x_a, a) = \frac{x_a - a_{\min}}{a_{\max} - a_{\min}} \tag{2.3}$$

and where $a_{\max}$ and $a_{\min}$ are the largest and smallest values yet processed for a.

Table 2.2: The IB1 testing algorithm.

Key: T: Test set
P: Set of predictor attributes
t: The target attribute
k: Number of most similar instances used
PCD: Partial concept description

Test(T, P, t, k)
1. for each $x_i \in T$ do
   1.1 $x_i \leftarrow$ Pre_process(x_i, P)
   1.2 return Performance(x_i, P, t, k)

Performance(x, P, t, k)
1. $S \leftarrow \emptyset$
2. $\forall y_i \in$ PCD:$S \leftarrow S \cup \{\langle y_i, \text{Similarity}(x, y_i, P)\rangle\}$
3. KSET $\leftarrow$ k_most_similar_instances(S, k)
4. return Target_value_prediction(KSET,t, k)

Learning: IB1's memory updating (learning) algorithm simply stores all processed training instances into the target's partial concept description.

IB1's testing algorithm is summarized in Table 2.2. IB1's behavior during testing differs from its behavior during training. IB1's testing algorithm is given the test set, the set of predictor attributes, the target attribute, and the value to be used for k. IB1 first pre-processes the test instance. The learning process for updating the numeric-valued predictor attributes observed minimum and maximum values is skipped during testing. Instead, the test instance is simply normalized by IB1's pre-processor. IB1 then calls on the performance component to compute the test instance's similarity to all processed training instances, selects the k most similar instances, and generates a target value prediction based on their target values.

Similarity: IB1's similarity function was informally described in the previous section, where it was not made obvious that the set of predictors used can vary depending on the target attribute. Therefore, IB1's similarity function is more appropriately described as a function of three inputs:

$$\text{Similarity}(x, y, P) = \frac{1}{\sqrt{\sum_{i=1}^{|P|} \text{Attribute\_difference}(x_i, y_i)}}, \tag{2.4}$$

where Attribute_difference is defined as

$$\text{Attribute\_difference}(x_i, y_i) = \begin{cases} \max(x_i - 0, 1 - x_i) & \text{if } y_i \text{ is missing} \\ \max(y_i - 0, 1 - y_i) & \text{if } x_i \text{ is missing} \\ 1 & \text{if both values are missing} \\ (x_i - y_i)^2 & i \text{ is numeric-valued} \\ x_i \neq y_i & \text{otherwise} \end{cases} \quad (2.5)$$

IB1 assumes that a missing attribute value is maximally different from the value present. If both values are missing for an attribute, then Attribute_difference yields 1. This algorithm for processing missing attribute values is evaluated in Section 5.2.1.

IB1's implementation is not specified. It could calculate similarities either iteratively or in parallel. The studies described in Chapter 4 concerning IB1's efficiency are independent of its implementation. More specifically, IB1's efficiency is defined as a function of the total number of attribute observations required for some (stated) purpose rather than in terms of elapsed cpu time. However, since similarity computations are independent, IB1's required cpu time could be significantly increased by evaluating them in parallel (Stanfill & Waltz, 1986; Waltz, 1990).

Prediction: IB1 uses a k-most similar instance decision rule to predict both symbolic and numeric values. More precisely, IB1 uses a simple majority vote function to determine the classification prediction for symbolic values; the predicted target value is the one that occurs most frequently among the k most similar training instances. Ties result in random selection among the set of most frequent target values. The prediction function for numeric target values computes a weighted-similarity of its k most similar instances' target values:

$$\text{Target\_value\_prediction(KSET,}t\text{,k)} = \sum_{i=1}^{k} \frac{\text{Similarity[KSET}_i\text{]} \times \text{KSET}_{i_t}}{\sum_{j=1}^{k} \text{Similarity[KSET}_j\text{]}}, \quad (2.6)$$

where KSET_i is one of the k most similar stored instances, KSET_{i_t} is instance KSET_i's value for target attribute t, and Similarity[KSET_i] is KSET_i's pre-computed similarity with the current test instance x_i. A majority vote function for predicting numeric values is not useful in many learning situations, namely those in which it is rare that two instances have the same value for the numeric-valued target attribute. This similarity-weighted function is more useful because it combines the predictions of similar instances based on their similarity to the instance given to the algorithm.

The setting for k can substantially affect IB1's learning behavior. Although the empirical investigations in Chapter 4 assume that $k = 1$, Section 5.2.2 describes an investigation in which k's setting is varied.

2.3.2 Example Applications of IB1

The accuracy of IB1's concept descriptions usually improves from modifications during the incremental training process. Suppose that the concept was *fast cars* and that the only available information on the car was its width and horsepower, although I will give unique names to each car according to its make. If the first two instances were

Instance Name	Width in Inches	Horsepower	Classification
Jaguar	70.6	262	Fast
Toyota	64.4	56	Slow

then they would be normalized by IB1 to have the values

Instance Name	Width in Inches	Horsepower	Classification
Jaguar	1	1	Fast
Toyota	0	0	Slow

Suppose that the third car has width 64.80, can generate 121 horsepower, and is considered to be fast. IB1 would normalize this car to have the values

Instance Name	Width in Inches	Horsepower	Classification
BMW	0.06	0.32	Fast

If asked to predict its classification when $k = 1$, then IB1 would misclassify this instance as a slow car because its normalized similarity to the Toyota

$$\text{Similarity(BMW,Toyota)} = \frac{1}{\sqrt{0.06^2 + 0.32^2}} = 3.03$$

is higher than to the Jaguar

$$\text{Similarity(BMW,Jaguar)} = \frac{1}{\sqrt{0.94^2 + 0.68^2}} = 0.86$$

However, IB1's concept description improves with training. If the BMW instance is stored with the previous two training instances, then some subsequently presented instances will be correctly rather than incorrectly classified. For example, if the next car had a width of 65.5 inches, generated 154 horsepower, and was considered to be fast, then the current set of normalized instances would be

Instance Name	Width in Inches	Horsepower	Classification
Jaguar	1	1	Fast
Toyota	0	0	Slow
BMW	0.06	0.32	Fast
Alfa Romero	0.18	0.48	Fast

and the new instance's similarities to the previous three instances would be

Instance Name	Similarity to the Alfa Romero
Jaguar	1.03
Toyota	1.95
BMW	5.00

Therefore, IB1 would classify the Alfa Romero according to the BMW's classification, which is *fast*. However, this instance would have been misclassified by the Toyota example if the BMW example had not yet been processed because the Toyota is more similar to the Alfa Romero than the Jaguar.

It is instructive to examine an extended pictorial example of IB1's behavior to visualize why these accuracy improvements occur. A visual description must display both the correct mapping of instances to target values and a summary of the predictions made by the current concept description. Accuracy improvements become obvious when comparing a sequence of figures that show this mapping at different points in time (i.e., after different numbers of training instances have been processed). For symbolic-valued targets, this improvement becomes vivid when a single target concept is defined by two numeric-valued dimensions, k is set to 1, and a *Voronoi diagram* (Voronoi, 1908; Shamos & Hoey, 1975; Bowyer, 1981; Watson, 1981; Seidel, 1987) is used to display the predictions made by IB1's concept description. Voronoi diagrams are useful for this purpose because they partition the instance space into regions, one for each stored instance, such that a unique stored instance in the partial concept description is most similar (among all stored instances) to every instance in its region. Since $k = 1$, the diagram describes which instances will be classified by each stored instance. The diagram would explicitly show the complete mapping of instances to symbolic-valued target values.

For example, Figure 2.3 shows how IB1's predictions change as it processes the first four instances in a training set. These instances were drawn randomly according to a uniform distribution from the two-dimensional space shown in each of the four snapshots of the instance space. The variable k (i.e., the number of most similar instances used to derive predictions) is set to 1 – it is far more difficult to generate detailed and clear descriptions of IB1's behavior for higher values of k. In this case, there is one symbolic concept to learn, which is defined by the four disjuncts delineated by the dashed lines in the snapshots. Instances in any of the disjuncts have a "+" target value; all others have a "–" value. The first instance happens to be positive. At this point, IB1 assumes that all instances are positive. It recorded a 27.8% classification accuracy on a separate test set of 1000 instances. After the second instance is processed, IB1's most similar neighbor prediction function predicts that all instance's closer to the first instance would be positive and all others are negative. The Voronoi diagramming line shown in the second snapshot shows this implicit partitioning of the instance space. IB1's accuracy increased dramatically after it processed the second instance. This was expected since the majority of the instances in this space would be classified as negative. The third instance, shown near the bottom of the

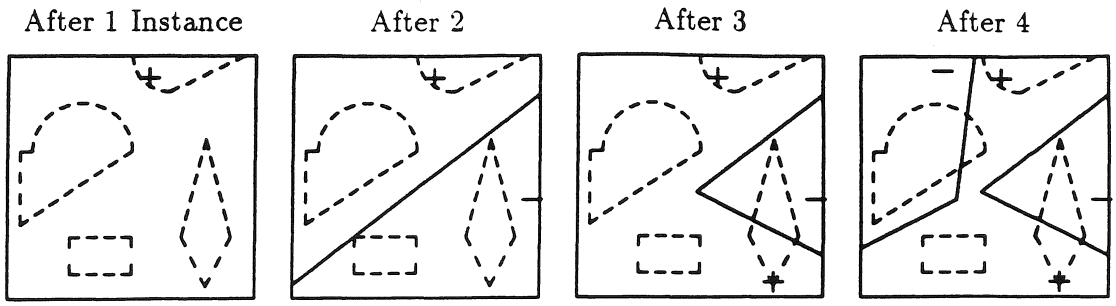


Figure 2.3: Voronoi diagrams describing IB1's classification predictions as the first four instances are processed. IB1's predictions, denoted by the solid lines, improves with training. Dashed lines delineate the four disjuncts of the target concept. The positive (+) and negative (−) training instances are shown. IB1's classification accuracy on a test set of 1000 instances is listed below each snapshot.

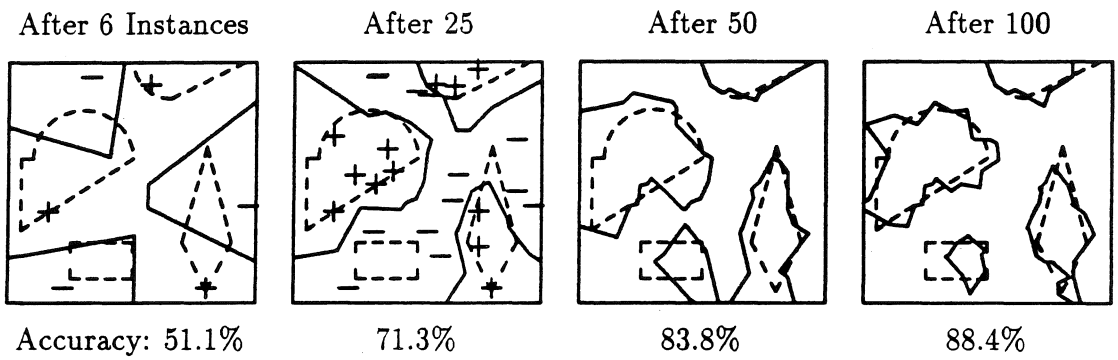


Figure 2.4: Voronoi diagrams describing IB1's classification predictions after 6, 25, 50, and 100 training instances have been processed. The training instances aren't shown in the latter two snapshots to reduce clutter. Classification accuracy, listed below the snapshots, improves with training.

third snapshot, introduced a third line segment into the diagram between the second and third instances (segments in the Voronoi diagram that delineate adjacent instances with the same target value are not shown). The size of the region predicted to contain only negative instances has decreased. Therefore, it is not surprising that IB1's classification accuracy decreased on the test set. Finally, the fourth instance increased the area predicted to contain negative instances, but simultaneously increased the area in the class's disjuncts that are also predicted to be negative. Accuracy increased only slightly. This process continues in Figure 2.4, which shows how the Voronoi diagram continues to improve its approximation's accuracy of the target concept as training continues. After 100 instances, it achieves an

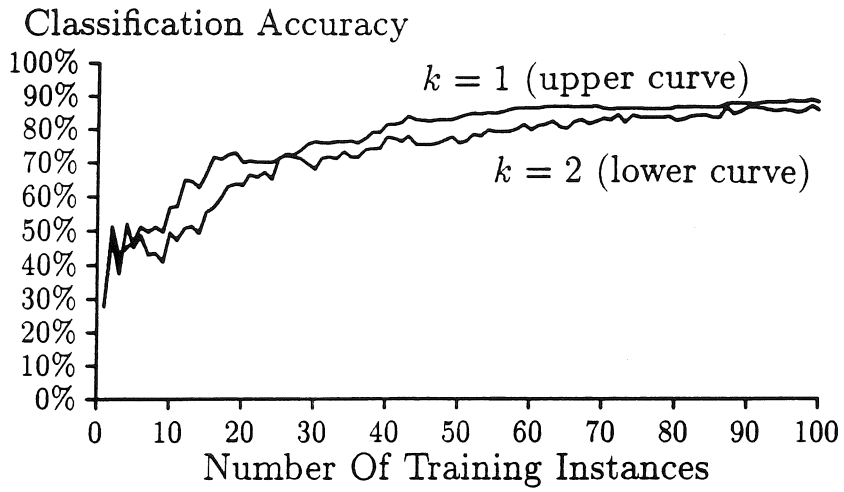


Figure 2.5: IB1's learning curves for predicting values for the symbolic concept shown in the previous figures. Its behavior is shown for two settings for k.

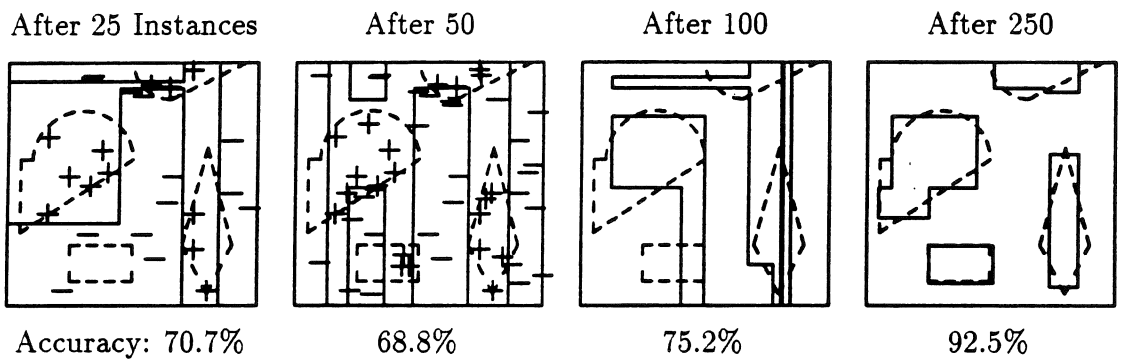


Figure 2.6: C4's extension eventually converges to an approximation similar to IB1's on this same training set, although C4's significance testing slows its learning rate.

88.4% classification accuracy on the same test set. IB1's learning curve for this example is detailed in Figure 2.5. This figure shows that, for this application, IB1's behavior when $k = 1$ is similar to its behavior when $k = 2$. I will detail the relative benefits for different values of k in Section 5.2.2.

It is interesting to compare IB1's snapshots with those generated by other learning algorithms to see how they differ. For example, the concept description learned by Quinlan's (1987) C4 decision tree algorithm yields the snapshots shown in Figure 2.6. It is somewhat unfair to compare IB1 to C4 here since the latter employs a significance test that slows

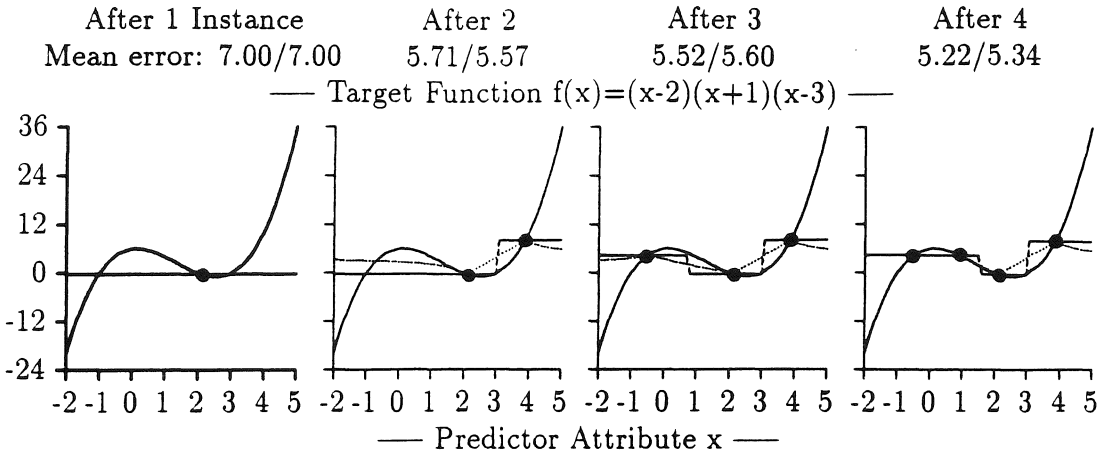


Figure 2.7: IB1's predictions for a simple cubic function (solid line) after processing the first 4 instances in the training set. The dashed line signifies IB1's predictions when $k = 1$ while the dotted line is for $k = 2$. IB1's mean error is shown above each snapshot for $k = 1$ and $k = 2$ (e.g., 5.71/5.57 indicates that 5.71 is $k = 1$'s mean error and 5.57 is $k = 2$'s). The test set contains the set of instances whose values for attribute x ranges from -2 to 5 by 0.1.

its learning rate.[4] My point here is that both algorithms will eventually converge to approximately the same concept description. Furthermore, the results described in Chapter 3 suggest that IBL and learning algorithms that build decision trees can PAC-learn the same class of concepts. However, I will show in Chapter 4 that their efficiency varies greatly depending on the characteristics of the application domain.

It is important to note that IBL algorithms do not reason from Voronoi diagrams. These diagrams reside only implicitly in the partial concept description as operated on by the similarity and prediction functions.

Voronoi diagrams are not useful for describing IB1's behavior as it learns to predict numeric values. Instead, its behavior can be observed more easily when its predicted values are shown periodically across the entire range of the target function. This can be described clearly when there is only one predictor attribute. For example, Figures 2.7 and 2.8 detail IB1's behavior as it learns to predict numeric values for a simple cubic function. The predictor attribute's values for the training instances were again drawn randomly according to a uniform distribution. Training instances are shown as circles in the eight snapshots in these

[4]C4 generates a complete decision tree for classifying all of the training instances. It then employs a test to determine which of the tree's nodes reflect statistically significant information. Nodes that don't pass this test are removed from the tree because they may represent noisy information. When no noise is present in the training instances, this *pruning* process results with fewer instances being used to generate the tree. Consequently, this reduces C4's learning rate.

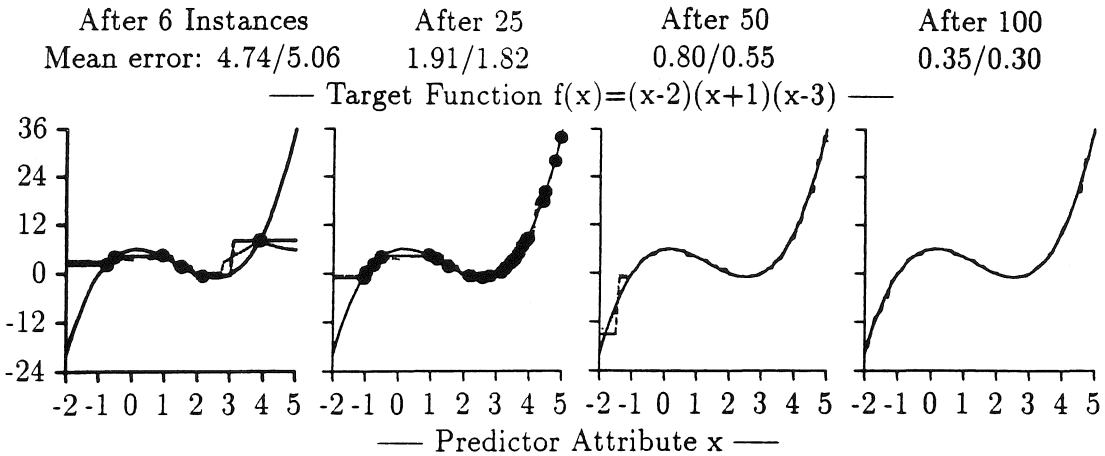


Figure 2.8: IB1's predictions after processing 6, 25, 50, and 100 instances respectively from the same training set.

figures (except where cluttering becomes a problem). As expected, IB1's approximation of the target function is poor until after several training instances have been processed. The mean errors were computed from a set of test instances whose values for the horizontal predictor attribute were spread evenly across its range with a resolution of 0.1. These errors decrease monotonically with the number of training instances. IB1's behavior here is slightly better when $k = 2$. However, the learning curves for this example, shown in Figure 2.9, shows that IB1's behavior is highly similar for these two settings of k.

2.3.3 Summary of IB1

IB1 is a simple algorithm that is amenable to mathematical analysis, which is the topic of the next chapter. However, it has high storage requirements (equal to the number of instances in the training set), which negatively impacts its efficiency. Section 4.2 examines an extension of IB1 that reduces its storage requirements. IB1 is also insensitive to the relative relevance of the predictor attributes used to describe instances. This causes problems since less relevant attributes should have less affect on prediction decisions. IB1 must learn these relative relevances and provide this information to its similarity and prediction functions so that its learning rate is not impaired by the introduction of additional irrelevant attributes. This problem is examined in Section 4.4. The next section surveys the performance dimensions that will be used to measure the capabilities of IBL algorithms in the rest of this dissertation.

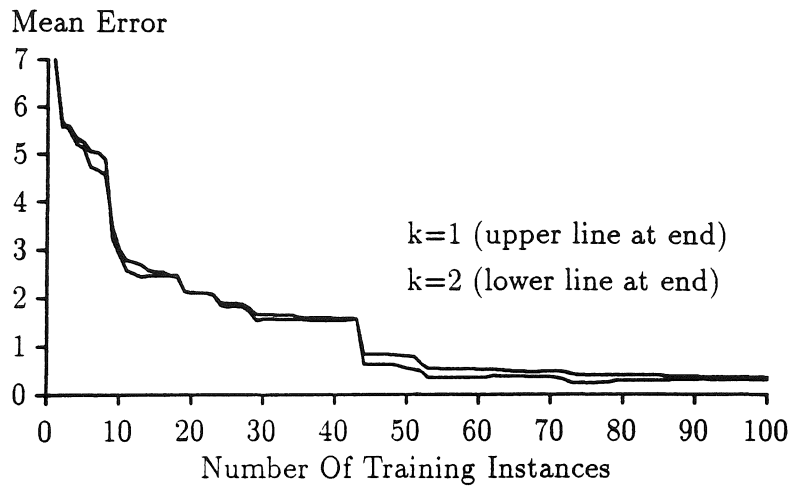


Figure 2.9: IB1's learning curves when $k = 1$ and $k = 2$. The numeric target function is $f(x) = (x - 2)(x + 1)(x - 3)$.

2.4 Performance Dimensions

Learning always takes place in the context of a particular purpose or set of purposes. IBL algorithms can be used to assist in solving several distinct problems, such as unsupervised learning (Stanfill, 1988) and learning from a helpful teacher (Salzberg, Delcher, Heath, & Kasif, 1990). I will focus on only one purpose in this dissertation: supervised learning from examples that are supplied by the external environment. This focus allows me to describe an exhaustive examination of IBL algorithms for this task instead of a brief treatment of several different tasks.

The capability of a learning algorithm can be measured by determining how well it satisfies its intended purpose(s). There are several good reasons to measure the performance of learning algorithms. First, they allow for comparisons between different learning algorithms targeted for satisfying the same purpose. System builders can then use these comparisons to predict more accurately which algorithms are useful for their purposes. Second, they describe the overall capabilities of learning algorithms for specific tasks. This highlights the limitations of currently existing approaches and suggests useful goals for further research. Finally, measurements can be used to refute or support (perhaps even validate) hypotheses that scientists want to state for a particular learning algorithm.

The next four chapters of this dissertation focus on measuring the performance of IBL algorithms along the following dimensions:

1. *Generality of Applicability*: This concerns the class of concepts that are describable by the representation and learnable by the algorithm. In Chapter 3, I will show that IBL algorithms can PAC-learn (Valiant, 1984) any symbolic concept whose boundary is a union of a finite number of closed hyper-curves of finite size and any continuous function with a bounded derivative.
2. *Resource Efficiency*: This encompasses a number of dimensions that measure the resources required to learn concepts that are learnable by the algorithm.
   (a) *Learning Rate*: This is the speed at which classification accuracy increases during training. It can be measured in many ways, including by recording the number of instances required to attain a given accuracy in an application domain. Alternatively, the number of attribute references can be counted. Learning rate is a more useful indicator of the performance of the learning algorithm than is accuracy for finite-sized training sets. It is often visually described using a learning curve.
   (b) *Accuracy*: This is the concept descriptions' prediction accuracy. It can be measured by testing on the training set itself (e.g., resubstitution estimates, recent training instances only, etc.) or by testing on a set of instances separate from the ones used during training (e.g., test sampling, cross-validation, etc.). I will use only the second method to measure accuracy.
   (c) *Processing Costs*: These are the costs involved with processing training instances. This includes a measure of the effort required to update concept descriptions, which is a relatively simple task for IBL algorithms but can be complex for algorithms that maintain large abstractions.
   (d) *Storage Requirements*: This refers to the size of the partial concept description. For IBL algorithms, this usually means the number of saved instances used for classification decisions.
3. *Psychological Plausibility*: This refers to the learning algorithm's ability to simulate behavior exhibited by humans (or, in some cases, other biological organisms). A psychologically plausible learning algorithm is one that exhibits many traits associated with humans and few others.

2.5 Chapter Summary

This chapter described a general framework for IBL algorithms, showed how a simple algorithm instantiates this framework, and summarized the set of dimensions that will be used to measure the performance of IBL algorithms in the remainder of this dissertation. In effect, this chapter sets the stage for Chapters 3 through 7 of this dissertation. Chapter 3 addresses the dimension of generality. It describes mathematical analyses for IB1. Proofs will

show that, for any value of k and any dimensionality, IB1 converges to a close approximation of the target concept with high probability for a large class of target concepts and functions with respect to the size of the training set. Chapter 4 describes three extensions of IB1 and evaluates their ability to reduce storage requirements, tolerate noisy training data, and learn relative attribute relevances. Chapter 5 then describes parametric studies of IB1's extensions and evaluates alternative instantiations for their framework component functions. Chapter 6 relates the IBL framework to cognitively plausible exemplar-based models of categorization. Finally, Chapter 7 reviews related IBL algorithms in terms of their contributions to the IBL framework.

Chapter Acknowledgements

Thanks to Dennis Kibler for encouraging me to develop a framework for instance-based learning algorithms, which was first discussed in (Aha, 1989a) and (Aha & Kibler, 1989). The updated version described in this chapter better reflects the typical separation of the performance and learning aspects of IBL systems. This framework helped me to coherently relate the IBL algorithms in the machine learning literature (see Section 7.2).

Chapter 3
Mathematical Analyses

In this chapter I will mathematically examine the dimension of generality to determine which assumptions guarantee that IBL algorithms will converge. My goal is to prove that, under certain restrictions, IBL algorithms require only a polynomial number of instances to learn a large class of concepts. However, the analyses presented in this chapter are limited. First, they assume that all predictor attributes are numeric-valued, although IBL algorithms can also process symbolic-valued predictor attributes. This is in itself not unusual; most analyses on the convergence properties of instance-based learning algorithms in the pattern recognition literature have ignored symbolic-valued attributes. Second, the following analyses yield only worst-case bounds for the number of examples required to learn concepts. Although average-case analyses can often be more useful, few researchers have been able to report progress on this difficult topic, and then only for relatively simple learning algorithms (e.g., Pazzani & Sarrett, 1990). Finally, these analyses are only for the IB1 algorithm, although they apply to all values for k (i.e., the number of most similar instances used to derive a prediction) and instance spaces of any dimensionality. More elaborate IBL algorithms are empirically evaluated in Chapter 4.

Marc Albert and Dennis Kibler were primarily responsible for developing the proofs described in this chapter, which are extensions of the ones described in (Kibler, Aha, & Albert, 1989) and (Aha, Kibler, & Albert, in press). The significant contributions of these mathematical analyses are that (1) the IB1 algorithm is proven to be applicable to a large class of concepts and numeric functions, (2) the upper bound on the number of instances that it requires to PAC-learn a target concept is proven to be polynomial in the size of the target concept's boundary, and (3) this bound is proven to be polynomial in the magnitude of the target function's slope for numeric prediction tasks.

The analysis is based on Valiant's (1984) probabilistic framework for the study of learning algorithms, which is known as the *PAC-learning* model. This model and related predecessor models for studying algorithms similar to IBL algorithms are described in Section 3.1. Section 3.2 then introduces definitions and lemmas that are used in the theorems presented in Section 3.3. Problems with analyzing other IBL algorithms, including instance-averaging algorithms (e.g., Sebestyen, 1962; Kohonen, 1986; Bradshaw, 1987), are discussed in Section 3.4.

3.1 Style of Analysis

This section briefly surveys previously reported mathematical analyses of IBL (and similar) algorithms, summarizes the Valiant (1984) model, motivates why this model was chosen to analyze IBL algorithms, and highlights how the following analyses depart from the customary use of Valiant's model.

Instance-based learning algorithms were derived from the nearest neighbor (NN) and k-nearest neighbor (k-NN) classification algorithms, which assign a classification to an instance based on the classifications of the most similar or set of k most similar previously processed instances. These classification algorithms are *nonparametric* (Fukunaga, 1972) in that they are applicable in situations when the distributions of instances is not restricted to a specific parametric family (e.g., unimodal). Nonparametric algorithms directly compute classification decisions rather than estimate parameter settings for known density functions and use them as the basis for classifications. This is useful for such nonparametric problems as those involving multimodal densities. These characteristics make NN and k-NN attractive for investigation.

The earliest published investigation on classification algorithms of the NN and k-NN variety was reported by Fix and Hodges (1951; 1952). They investigated the performance of a k-NN type of algorithm for both infinite- and small-sized training sets. Subsequently, several other researchers noted the appeal of the nearest neighbor algorithm. However, the first analytic results in the nonparametric case, for either infinite- or finite-sized training sets, was reported by Cover and Hart (1967). They were interested in evaluating the accuracy of NN in comparison with all other classification algorithms. Given complete information concerning the joint probability densities of all instances in an instance space, the optimal strategy for classifying instances is the Bayes decision strategy. That is, this strategy, which is described in more detail in Section section:relation-pattern-recognition, minimizes the probability of classification error. Cover and Hart showed that NN's probability of classification error was less than twice that of the Bayes optimal strategy. Therefore, it is less than twice the error rate of *any* decision algorithm. Subsequently, Duda and Hart (1973) showed that, with increasing odd values for k, k-NN's error rate quickly approaches the optimal Bayes rate, becoming identical with it when k reaches infinity.

Although these results show that NN is a competent classifier, they assumed that the number of stored instances was infinite, which rarely occurs in practice. Also, previous investigations of these classification algorithms did not determine how many instances are required to assure that these classification algorithms will yield acceptably good classification decisions. Hart (1968) posed this question to his readers after his introduction of the *condensed nearest neighbor* (CNN) algorithm. The analyses described in this chapter are used to answer his question in Section 4.2.

Our analysis employs Valiant's (1984) probabilistic model for investigating learnability issues. This model states that a class of concepts is *polynomially learnable* from examples if at most a polynomial number of instances[1] is required to generate, with a certain level of confidence, a relatively accurate approximation of the target concept. This definition of learnability is more relaxed than the definitions used in most earlier studies, which required that the generated concept descriptions be completely accurate with 100% confidence.

The classes of concepts most frequently studied using Valiant's model have involved Boolean formulae of n variables (e.g., DNF, CNF, decision lists, and their variants). The target attribute was assumed to have two values – *positive* and *negative* – corresponding to concept members and nonmembers respectively. This model does not specify learning algorithms for learnable concept classes. It only confirms whether there exists a learning algorithm such that, under any fixed probability distribution for the instances, can learn a good approximation of any concept in a given concept class in polynomial time with high confidence. Definition 3.1 formalizes this notion of learnability.

Definition 3.1 A class of concepts $\mathcal{C}$ is *polynomially learnable* iff there exists a polynomial p and a learning algorithm A such that, for any $0 < \epsilon, \delta < 1$, if at least $p(\frac{1}{\epsilon}, \frac{1}{\delta})$ positive and negative instances of a concept $C \in \mathcal{C}$ are drawn according to an arbitrary fixed distribution, A will generate, with confidence at least $(1 - \delta)$, an approximation of C whose probability of error is no more than ϵ. Moreover, A will halt in time bounded by a polynomial in the number of instances.

This definition states that the approximation generated does not have to be perfect, but only with inaccuracy bounded by ϵ. Also, A is not required to always generate sufficiently accurate concept descriptions (i.e., within ϵ), but must do so only with probability at least $(1 - \delta)$. Thus, this is called the *PAC*-learning model (**probably, approximately correct**). Finally, the concept class $\mathcal{C}$ and the space of hypotheses generated by A are not required to be identical. For example, Pitt and Valiant (1986) showed that k-term-DNF is learnable by k-CNF.

Many classes of concepts are PAC-learnable when $\mathcal{C}$ matches the space of hypotheses generated by A. For example, Valiant (1984) showed that the class of concepts representable by k-CNF (for all k) is PAC-learnable from positive examples only, Rivest (1987) showed that the class of k-decision lists is PAC-learnable, and Haussler (1986) showed that the class of internal disjunctive formulae is PAC-learnable. Many proofs have also been published that show which classes of concepts are *not* PAC-learnable (Pitt & Valiant, 1986; Kearns, Li, Pitt, & Valiant, 1987a). Finally, Blumer, Ehrenfeucht, Haussler, and Warmuth (1989) extended Valiant's model to learning classes of concepts whose predictor attributes are numeric-valued.

Valiant's model for analysis trades off accuracy for generality. It is extremely general; it applies to any fixed distribution. Thus it is said to be *distribution-free*. However, the number

[1]That is, polynomial with respect to the parameters for confidence and accuracy.

of instances required to PAC-learn concepts is a loose upper bound that could be tightened, for example, by placing restrictions on the set of probability distributions to be analyzed. That is, this learnability model is frequently used to determine how many instances are required to PAC-learn a class of concepts $\mathcal{C}$, but the result is indicative of only the concept in $\mathcal{C}$ that is most difficult to learn and the probability distribution that results in the slowest possible learning rate. Also, it does not specify *which* algorithm can PAC-learn a class of concepts, but only that one exists (although some authors do present an example algorithm that can PAC-learn the target class of concepts).

Therefore, many researchers have adapted the model to their own needs by adding restrictions or analyzing a specific set of learning algorithms. For example, Li and Vitanyi (1989) showed that several more general classes of concepts are learnable under Valiant's model once the *distribution-free* assumption is replaced with a *simple distribution* assumption. Kearns, Li, Pitt, and Valiant (1987b) have also discussed distribution-specific results. Littlestone (1988) used Valiant's model to analyze a specific learning algorithm, one that incrementally learns Boolean functions. Similarly, Valiant (1985), Haussler (1987), Rivest (1987), and Angluin and Laird (1988), have also examined specific learning algorithms using this model.

The analyses presented in this chapter enforce similar restrictions on the use of the PAC-learning model. In particular, restrictions are placed on the set of allowable probability distributions of the instances in the instance space. Also, the analyses investigate the capabilities of a specific learning algorithm, IB1, rather than focus on the learnability of concept classes in general. Finally, the predictor attributes are assumed to be numeric-valued and noise-free.

Section 3.3.2 describes which classes of numeric functions are learnable by IB1. Few researchers have investigated the learnability of numeric functions. This is one of the more novel contributions of this chapter to the literature on computational learning theory.

With these restrictions in place, the proofs in Section 3.3 show that IB1 can learn a class of concepts with an infinite *Vapnik-Chervonenkis (VC) dimension* (Vapnik & Chervonenkis, 1971). The VC dimension of a class of concepts is explained by the following two definitions.

Definition 3.2 A set $S \subset X$ of instances in an instance space X is *shattered* by a class $\mathcal{C}$ of concepts if, for every subset $U \subset S$ of instances, there exists a concept description $C \in \mathcal{C}$ such that C yields *positive* for instances in U and *negative* for instances in $S - U$.

That is, a class $\mathcal{C}$ of concepts shatters a set X of instances if, for each subset $S \subset X$, a member of $\mathcal{C}$ perfectly classifies instances according to whether or not they are in S.

Definition 3.3 The *VC dimension* of a non-empty class of concepts $\mathcal{C}$ is the cardinality of the largest set that is shattered by $\mathcal{C}$.

Blumer and his colleagues (Blumer *et al.*, 1986) proved that a concept class C is learnable with respect to the class of all probability distributions iff C has a finite VC dimension. The theorems in Section 3.3.1 show that IB1 can learn the class of concepts describable by a finite union of closed hyper-curves of finite size, which has infinite VC dimension. This apparent contrary result is easily explained. Whereas Blumer et al.'s result is distribution-free, the theorems in Section 3.3.1 place restrictions on the class of allowable probability distributions.

3.2 Coverage Lemmas

This section details four lemmas that are used to help prove the theorems described in Section 3.3, which concern the polynomial-time learning capabilities of IB1. These lemmas establish the number of training instances in a training set S that are required to give, with high confidence, a "good coverage" of an instance space. That is, they ensure that, for all instances x in the space, except for those in regions of low probability, there is an instance (or set of k instances) in S that is sufficiently similar to x (i.e., their similarity is above a threshold).[2] This information will be used in the learnability theorems to put a bound on IB1's number (or amount) of prediction errors.

These lemmas differ in terms of their generality. The first lemma applies to instance spaces with one dimension, the second applies to spaces with two dimensions, and the third applies to spaces with any number of dimensions. The fourth lemma extends the third to any value for k (i.e., the number of most similar instances to use for predictions of target values). All of the predictor attribute dimensions are assumed to be numeric-valued.

The set of variables used in the definitions, lemmas, and proofs are summarized in Table 3.2. The first two variables are normally used in the PAC-learning model. The other variables refer to the notion of $\langle \alpha, \gamma \rangle$-nets, which are introduced in the following two definitions and are used in the proofs of the lemmas and theorems in this chapter.

Definition 3.4 The *α-ball* about an instance x in $\Re^d$ is the set of instances within distance α of x: $\{y \in \Re^d | \text{distance}(x, y) < \alpha\}$.

Definition 3.5 Let X be a subset of $\Re^d$ with an arbitrary but fixed probability distribution. A subset $S \subseteq X$ is an *$\langle \alpha, \gamma \rangle$-net* for X if, for all x in X, except for a set $S_{\geq \alpha}$ with probability less than γ, there exists an $s \in S$ such that $\text{distance}(s, x) < \alpha$. Figure 3.2 displays an example $\langle \alpha, \gamma \rangle$-net for the unit interval domain, where α is set to 0.1.

The definition of an $\langle \alpha, \gamma \rangle$-net applies to any probability distribution. There are many distributions such that, if a large set of instances was generated according to the distribution,

[2] In this chapter, I will assume that similarity is defined as the inverse of Euclidean distance.

Table 3.1: Key to some of the variables used in the proofs.

Name	Description
ϵ (tolerance)	The upper bound on the probability of prediction error.
δ (confidence)	$(1-\delta)$ is the lower bound on the probability that the generated concept description's probability of prediction error is less than ϵ.
S	A sample set, or training set, from an instance space. It will always be assumed to be an $\langle\alpha,\gamma\rangle$-net.
$S_{<\alpha}$	The set of instances in the instance space that are guaranteed to be within distance α of some instance in S.
$S_{\geq\alpha}$	The set of instances in the instance space that are not guaranteed to be within distance α of some instance in S.
α (raw error)	The upper bound on the distance between an instance in $S_{<\alpha}$ and an instance in S.
γ	The upper bound on the probability that an arbitrary instance in the instance space is in $S_{<\gamma}$.

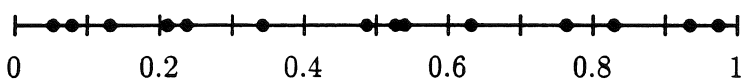


Figure 3.1: An example $\langle\alpha,\gamma\rangle$-net for the unit interval where $\alpha = 0.1$ and $\gamma = 0$. The training instances are marked by the black circles. Every instance along the interval is within 0.1 of some training instance.

some of the instances in the instance space would have no highly similar neighbors in the generated set. This set of "lonely" instances will always be referred to as $S_{\geq\alpha}$.

The following proof shows that a sufficiently large random sample from the unit interval will probably be an $\langle\alpha,\gamma\rangle$-net.

Lemma 3.1 *Let $0 < \alpha, \gamma, \delta < 1$. Then there exists a polynomial p such that a random sample S containing $N > p(\frac{1}{\alpha}, \frac{1}{\gamma}, \frac{1}{\delta})$ instances from $[0,1]$, drawn according to any fixed probability distribution, will form an $\langle\alpha,\gamma\rangle$-net with confidence at least $(1-\delta)$.*

Proof 3.1 Let α, γ, and δ be arbitrary positive numbers less than 1 and let P be the fixed distribution of the instances. We prove this lemma by partitioning the unit interval into m equal-lengthed sub-intervals (i.e., the sub-intervals are disjoint and their union exhausts the unit interval). Let $\alpha > \frac{1}{m}$. This ensures that the every instance in a sub-interval is within α of every other instance in that sub-interval because the length of each sub-interval is $\frac{1}{m}$. However, the proof is simplified, without loss of generality, if we set $\alpha = \frac{1}{m}$.

The probability of a sub-interval is the probability that a randomly-selected instance in the unit interval, drawn according to P, lies in it. By selecting enough instances we assure that, with high probability, each sub-interval of sufficient probability will contain at least one of the selected instances. Let $S_{<\alpha}$ be the set of sub-intervals $[x_i, x_{i+1}]$ such that $Pr([x_i, x_{i+1}]) \geq \frac{\gamma}{m}$. These are the sub-intervals that will contain at least one instance from the sample. Let $S_{\geq\alpha}$ be the set of the remaining sub-intervals. Note that the summed probability of the sub-intervals in $S_{\geq\alpha}$ is $Pr(S_{\geq\alpha}) < m \times \gamma/m = \gamma$.

Now we want to determine a lower bound for N, the number of sample instances required to ensure, with probability at least $(1-\delta)$, that there is a sample instance in each sub-interval in $S_{<\alpha}$. The probability that an arbitrary instance $i \in [0,1]$ selected according to P will not lie in some selected sub-interval in $S_{<\alpha}$ is at most $(1 - \frac{\gamma}{m})$. The probability that none of the N sample instances will lie in a selected sub-interval in $S_{<\alpha}$ is at most $(1 - \frac{\gamma}{m})^N$. The probability that any sub-interval in $S_{<\alpha}$ is excluded by all N sample instances is at most $m(1 - \frac{\gamma}{m})^N$.

At this step of the proof, it is convenient to reformulate this expression as a simpler expression of at least equal value. In doing so, the upper bound on this probability will be loosened, but the comprehensibility of the eventual result is substantially improved. Specifically, I will show that

$$m(1 - \frac{\gamma}{m})^N < me^{\frac{-N\gamma}{m}}.$$

The steps proceed as follows:

$$\begin{aligned}
m(1 - \tfrac{\gamma}{m})^N &\stackrel{?}{<} me^{\frac{-N\gamma}{m}} \\
(1 - \tfrac{\gamma}{m})^N &\stackrel{?}{<} e^{\frac{-N\gamma}{m}} \\
\ln(1 - \tfrac{\gamma}{m})^N &\stackrel{?}{<} \ln e^{\frac{-N\gamma}{m}} \\
N\ln(1 - \tfrac{\gamma}{m}) &\stackrel{?}{<} \frac{-N\gamma}{m} \\
\ln(1 - \tfrac{\gamma}{m}) &\stackrel{?}{<} \frac{-\gamma}{m} \\
\ln(1 - \tfrac{\gamma}{m}) = \frac{-\gamma}{m} - \frac{1}{2}(\frac{\gamma}{m})^2 - \frac{1}{3}(\frac{\gamma}{m})^3 - \cdots &< \frac{-\gamma}{m}.
\end{aligned}$$

This probability can be forced to be small by setting it to be less than δ. N is then solved for as follows:

$$\begin{aligned}
me^{\frac{-N\gamma}{m}} &< \delta \\
e^{\frac{-N\gamma}{m}} &< \frac{\delta}{m} \\
-\frac{N\gamma}{m} &< \ln(\frac{\delta}{m})
\end{aligned}$$

$$
\begin{aligned}
N &> -\frac{m}{\gamma}\ln(\frac{\delta}{m}) \\
N &> \frac{m}{\gamma}\ln(\frac{m}{\delta}) \\
N &> \frac{1}{\alpha\gamma}\ln(\frac{1}{\alpha\delta})
\end{aligned}
$$

Consequently, with probability at least $(1-\delta)$, each sub-interval in $S_{<\alpha}$ contains some sample instance. Also, the total probability of all the sub-intervals in $S_{\geq\alpha}$ is less than $\frac{\gamma}{m}m = \gamma$. Since each instance of $[0,1]$ is therefore in some sub-interval of $S_{<\alpha}$, except on a set of probability less than γ, then, with probability at least $(1-\delta)$, an arbitrary instance of $[0,1]$, selected according to the given probability distribution, is within α of some sample instance (except on a set of probability less than γ). ■

Since all attribute dimensions are normalized by IBL algorithms, this proof immediately generalizes to all bounded one-dimensional instance spaces whose predictor attribute is numeric-valued.

Lemmas 3.2 and 3.1 are identical except that the latter applies to two-dimensional spaces. It is detailed here to convince the reader that the coverage lemma will extend to $\Re^d$, which is later addressed by Lemma 3.3.

Lemma 3.2 *Let α, δ, and γ be fixed positive numbers less than 1. There exists a polynomial p such that a random sample S containing $N > p(\frac{1}{\alpha},\frac{1}{\gamma},\frac{1}{\delta})$ instances from [0–1,0–1], drawn according to any fixed probability distribution, will form an $\langle\alpha,\gamma\rangle$-net with probability at least $(1-\delta)$.*

Proof 3.2 This lemma is proven by partitioning the unit square into m^2 disjoint sub-squares of equal area. Let the diagonal of each sub-square have length less than α. This ensures that all pairs of instances in the sub-square are within distance α of each other. This proof ensures that, with high confidence, at least one of the N sample instances lies in each sub-square of sufficient probability.

Let $m = \lceil\sqrt{2}/\alpha\rceil$. (We assume, without loss of generality, that $\lceil\sqrt{2}/\alpha\rceil > \sqrt{2}/\alpha$.) Let $S_{<\alpha}$ be the set of sub-squares with probability greater than or equal to γ/m^2. Let $S_{\geq\alpha}$ be the set of remaining sub-squares, which will therefore have summed probability less than $\frac{\gamma}{m^2}m^2 = \gamma$. The probability that an arbitrary instance $i \in$ [0–1,0–1] will not lie in some selected sub-square in $S_{<\alpha}$ is at most $1-\gamma/m^2$. The probability that none of the N sample instances will lie in a selected sub-square in $S_{<\alpha}$ is at most $(1-\gamma/m^2)^N$. The probability that any sub-square in $S_{<\alpha}$ is excluded by all N sample instances is at most $m^2(1-\gamma/m^2)^N$.

As in Lemma 3.1, it is convenient to reformulate this expression by using a simpler one with a higher value, even though this loosens the upper bound we will derive for the number of instances required to yield a $\langle \alpha, \gamma \rangle$-net. In this case, we want to show that

$$m^2(1 - \gamma/m^2)^N < m^2 e^{-N\gamma/m^2}.$$

This can be shown as follows:

$$\begin{aligned}
m^2(1 - \frac{\gamma}{m^2})^N &\stackrel{?}{<} m^2 e^{\frac{-N\gamma}{m^2}} \\
(1 - \frac{\gamma}{m^2})^N &\stackrel{?}{<} e^{\frac{-N\gamma}{m^2}} \\
\ln(1 - \frac{\gamma}{m^2})^N &\stackrel{?}{<} \ln e^{\frac{-N\gamma}{m^2}} \\
N \ln(1 - \frac{\gamma}{m^2}) &\stackrel{?}{<} \frac{-N\gamma}{m^2} \\
\ln(1 - \frac{\gamma}{m^2}) &\stackrel{?}{<} \frac{-\gamma}{m^2} \\
\ln(1 - \frac{\gamma}{m^2}) = \frac{-\gamma}{m^2} - \frac{1}{2}(\frac{\gamma}{m^2})^2 - \frac{1}{3}(\frac{\gamma}{m^2})^3 - \cdots &\stackrel{?}{<} \frac{-\gamma}{m^2}
\end{aligned}$$

This probability can be forced to be small by setting it to be less than δ. N can then be solved for as follows:

$$\begin{aligned}
m^2 e^{\frac{-N\gamma}{m^2}} &< \delta \\
e^{\frac{-N\gamma}{m^2}} &< \frac{\delta}{m^2} \\
\frac{-N\gamma}{m^2} &< \ln \frac{\delta}{m^2} \\
N &> \frac{m^2}{\gamma} \ln \frac{m^2}{\delta} \\
N &> \frac{\lceil \frac{\sqrt{2}}{\alpha} \rceil^2}{\gamma} \ln \frac{\lceil \frac{\sqrt{2}}{\alpha} \rceil^2}{\delta}
\end{aligned}$$

Consequently, with probability at least $(1 - \delta)$, each sub-square in $S_{<\alpha}$ contains some sample instance of S. Also, the total probability of all the sub-squares in $S_{\geq\alpha}$ is less than $(\gamma/m^2)m^2 = \gamma$. Since each instance of [0–1,0–1] is in some sub-square of $S_{<\alpha}$, except for a set of probability less than γ, then, with confidence at least $(1 - \delta)$, an arbitrary instance of [0–1,0–1] is within α of some instance of S (except for a set of small probability). ■

This proof extends to any bounded two-dimensional space. Lemma 3.3 emerges from the pattern seen in these first two lemmas.

Lemma 3.3 *Let α, δ, and γ be fixed positive numbers less than 1. There exists a polynomial p such that a random sample S containing $N > p(\frac{1}{\alpha}, \frac{1}{\gamma}, \frac{1}{\delta})$ instances from a bounded subspace of $\Re^d$ drawn according to any fixed probability distribution, will form an $\langle \alpha, \gamma \rangle$-net with probability at least $(1 - \delta)$.*

Proof 3.3 Similar to the proofs for Lemmas 3.1 and 3.2. In this case, it can be shown that

$$N > \frac{\lceil \frac{\sqrt{d}}{\alpha} \rceil^d}{\gamma} \ln \frac{\lceil \frac{\sqrt{d}}{\alpha} \rceil^d}{\delta}$$

■

Since IBL algorithms normalize the given attribute dimensions, Lemma 3.3 applies to any d-dimensional instance space.

The first three lemmas assume that the value of k is one. That is, they assumed that only the most similar stored instance is used to derive predictions of target values. The following lemma extends Lemma 3.3 to $k \geq 1$. The proof works by ensuring that there are at least k training instances in each sub-region of the partition. This will require that the confidence, represented by $(1 - \delta)$, decreases with increasing values of k.

Lemma 3.4 requires an extension to the definition of an $\langle \alpha, \gamma \rangle$-net.

Definition 3.6 Let X be a subset of $\Re^d$ with an arbitrary but fixed probability distribution. A subset $S \subseteq X$ is an k-$\langle \alpha, \gamma \rangle$-*net* for X if, for all $x \in X$, except for a set $S_{\geq \alpha}$ with probability less than γ, there exists at least k instances $s \in S$ such that distance$(s, x) < \alpha$.

Lemma 3.4 *Let α, δ, and γ be fixed positive numbers less than 1. There exists a polynomial p such that a random sample S containing $N > p(\frac{1}{\alpha}, \frac{1}{\gamma}, \frac{1}{\delta})$ instances from $\Re^d$ drawn according to any fixed probability distribution, will form a k-$\langle \alpha, \gamma \rangle$-net with probability at least $(1 - \delta)$.*

Proof 3.4 This proof ensures that, with high confidence, at least k of the N sample instances lies in each hyper-square of sufficient probability.

This lemma is proven by partitioning a bounded subset of $\Re^d$ into m^d disjoint hyper-squares of equal area, each with diagonal (i.e., the line segment joining the two most distant instances in the hyper-square) less than α. This ensures that all pairs of instances in each hyper-square are within distance α of each other.

The value for m is found using the Pythagorean Theorem, which shows that $m = \frac{\lceil \sqrt{d} \rceil}{\alpha}$. (We assume, without loss of generality, that $\lceil \sqrt{d}/\alpha \rceil > \sqrt{d}/\alpha$.) Let $S_{<\alpha}$ be the subset of hyper-squares with probability greater than or equal to γ/m^d. Let $S_{\geq \alpha}$ be the set of remaining hyper-squares, which will therefore have summed probability less than $\frac{\gamma}{m^d} m^d = \gamma$.

The probability that arbitrary instance i in a bounded subset of $\Re^d$ will not lie in some selected hyper-square in $S_{<\alpha}$ is at most $1 - \gamma/m^d$. The probability that none of the N sample instances will lie in a selected hyper-square in $S_{<\alpha}$ is at most $(1 - \gamma/m^d)^N$. The probability that any hyper-square in $S_{<\alpha}$ is excluded by all N sample instances is at most $m^d(1 - \gamma/m^d)^N$.

As in the previous lemmas, it is convenient to reformulate this expression by using a simpler one with a higher value, even though this loosens the upper bound to be derived for the number of instances required to form a k-$\langle\alpha, \gamma\rangle$-net. In this case, it can be shown that

$$m^d(1 - \gamma/m^d)^N < m^d e^{-N\gamma/m^d}.$$

This probability can be forced to be small by setting it to be less than δ'. The value of δ will later be solved for in terms of δ'.

N can be solved for as in the previous lemmas. The resulting inequality is

$$N > \frac{\lceil \frac{\sqrt{d}}{\alpha} \rceil^d}{\gamma} \ln \frac{\lceil \frac{\sqrt{d}}{\alpha} \rceil^d}{\delta'}.$$

Consequently, with probability at least $(1 - \delta')$, each hyper-square in $S_{<\alpha}$ contains some sample instance of S. Also, the total probability of all the sub-squares in $S_{\geq\alpha}$ is less than $(\gamma/m^d)m^d = \gamma$. Since each instance of a bounded subset of $\Re^d$ is in some hyper-square of $S_{<\alpha}$, except for a set of probability less than γ, then, with confidence at least $(1 - \delta')$, an arbitrary instance of this bounded subset is within α of some instance of S (except for a set of small probability).

This process needs to be repeated k times. In each case, the set $S_{\geq\alpha}$ is the same since the probability distribution (and, therefore, the set of hyper-squares with low probability) is the same. This yields the following inequality for assuring that k training instances are within α distance of each instance in $S_{<\alpha}$:

$$N > \frac{k\lceil \frac{\sqrt{d}}{\alpha} \rceil^d}{\gamma} \ln \frac{\lceil \frac{\sqrt{d}}{\alpha} \rceil^d}{\delta'}$$

Finally, δ' must be solved for in terms of δ. This can be done as follows:

$$\begin{aligned} (1 - \delta')^k &= 1 - \delta \\ 1 - \delta' &= \sqrt[k]{1 - \delta} \\ 1 - \sqrt[k]{1 - \delta} &= \delta' \end{aligned}$$

Substituting this expression for δ', this yields

$$N > \frac{k\lceil \frac{\sqrt{d}}{\alpha} \rceil^d}{\gamma} \ln \frac{\lceil \frac{\sqrt{d}}{\alpha} \rceil^d}{1 - \sqrt[k]{1 - \delta}}$$

■

Thus we are guaranteed that, by picking enough random samples, we will probably get a good coverage of any domain. However, the number of samples required to yield k-$\langle\alpha,\gamma\rangle$-nets (where $k \geq 1$) increases exponentially with the dimensionality of the instance space. This highlights the first problem addressed in Chapter 4: storage reduction. Methods for reducing the perceived dimensionality of the instance space are detailed in Section 4.4.

3.3 Convergence Theorems

This section details theorems that show that IB1 can learn a large class of concepts and numeric functions in polynomial time. However, there are several classes of concepts that IB1 will fail to learn. For example, it is unable to learn target concepts whose predictor attributes are logically inadequate for describing it (e.g., the concept of even numbers given positive and negative instances whose only attribute is their integer value). The proofs in this section show that IB1 can PAC-learn two restricted but general classes of concepts.

First, Section 3.3.1 describes theorems concerning IB1's ability to predict symbolic values. The proofs make a statistical assumption on the distributions of training sets, which requires a small extension of the PAC-learning model. They also make *geometric* assumptions to constrain the target concept. Rosenblatt (1962) demonstrated that if a concept is an arbitrary hyper-half-plane (i.e., the set of instances on one side of a hyper-plane), then the perceptron learning algorithm is guaranteed to converge. The proofs analyzing IB1's ability to predict symbolic values use a more general geometric assumption: they prove that the class of concepts describable by a finite union of closed hyper-curves of finite size is PAC-learnable by IB1. This may be the same requirement C4 (Quinlan, 1987), PLS (Rendell, 1988), and back-propagation (Rumelhart *et al.*, 1987) need to ensure that target concepts are learnable. Although they may have certain restrictions concerning the shape of their concept descriptions (e.g., a concept description formed by a decision tree that uses only single-attribute splits must consist of a union of hyper-rectangles), any concept description formed by an IBL algorithm can be closely approximated by a concept description generated by any of these algorithms.

Section 3.3.2 details the theorems concerning IB1's ability to predict numeric values. The corresponding proofs show that IB1 can learn the class of continuous, real-valued numeric functions with bounded slope in polynomial time. The PAC-learning model has rarely been used to address the issue of learning numeric-valued functions. Therefore, a substantially different definition of polynomial learnability is used in Section 3.3.2, although it preserves the spirit of Valiant's original model.

3.3.1 Convergence Theorems: Predicting Symbolic Values

This section details convergence theorems for the IB1 algorithm. The following extension of the definition for polynomial learnability will be used.

Definition 3.7 A class of concepts $\mathcal{C}$ is *polynomially learnable* from examples with respect to a class of probability distributions $\mathcal{P}$ iff there exists an algorithm A and a polynomial p such that for any $0 < \epsilon, \delta < 1$, and any $C \in \mathcal{C}$, if more than $p(\frac{1}{\epsilon}, \frac{1}{\delta})$ examples are chosen according to any fixed probability distribution $P \in \mathcal{P}$, then, with confidence at least $(1 - \delta)$, A will output a hypothesis for C that is a member of $\mathcal{C}$ which differs from C on a set of instances with probability less than ϵ.

This definition differs from Valiant's (1984) original model in that the class of allowable probability distributions $\mathcal{P}$ is constrained.

Theorem 3.1 describes the relationship, for a particular class of concepts, between a target concept C and the concept description approximation C' converged on by IB1 when the value of k is 1 and the dimensionality d is 2. Theorem 3.2 repeats Theorem 3.1 with $k \geq 1$ and arbitrary d. Finally, Theorem 3.3 adds constraints so that these theorems also imply convergence in terms of the class of allowable probability distributions. This allows us to show that IB1 can PAC-learn a large class of concepts, namely those describable as the finite union of finite-sized regions bounded by closed hyper-curves with finite hyper-length.

A few more definitions are needed for the analysis. Figure 3.3.1 illustrates these definitions.

Definition 3.8 For any $\alpha > 0$, let the *α-core* of a set C be all those instances of C such that the α-ball about them is contained in C.

Definition 3.9 The *α-neighborhood* of C is the set of instances that are within α of some instance of C.

Definition 3.10 The set of instances C' is an *$\langle \alpha, \gamma \rangle$-approximation* of C if, ignoring some set $S_{\geq \alpha}$ with probability less than γ, it contains the α-core of C and is contained in the α-neighborhood of C.

Note that, if the α-neighborhood of a set of instances contains the entire space, then that set is an $\langle \alpha, 0 \rangle$-net for this space.

The following theorem describes how accurately IB1's derived concept description approximates the target concept. In particular, the IB1 algorithm converges (with probability at least $(1 - \delta)$) to a concept which, except for a set of instances with probability less than γ, lies between the α-core and the α-neighborhood of the target concept. Theorem 3.1 then shows that IB1 can polynomially learn any concept describable as a union of any regions bounded by a closed curve in the unit square [0–1,0–1].

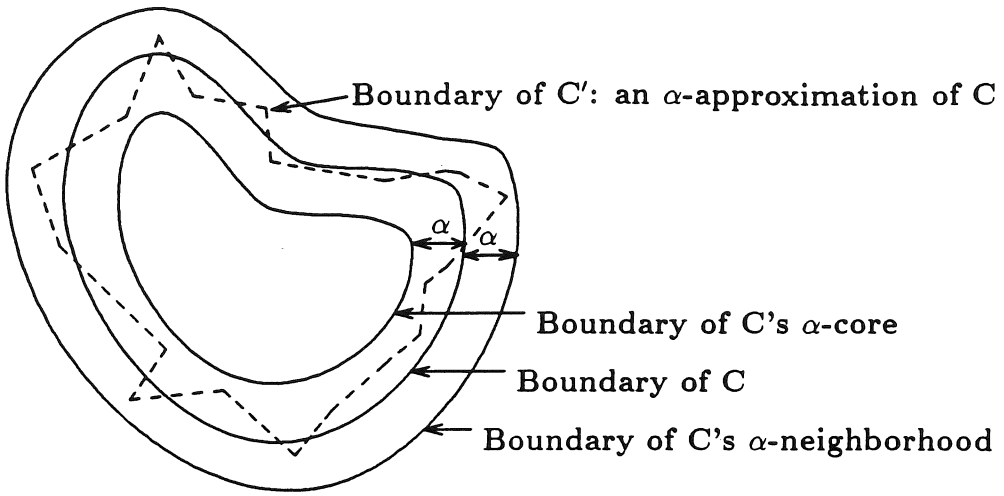


Figure 3.2: Exemplifying some terms used for analyzing learnability. This instance space has two numeric-valued attributes.

Theorem 3.1 *Let C be any region bounded by a closed curve in [0–1,0–1]. Given $0 < \alpha, \delta, \gamma < 1$, then the IB1 algorithm with $k = 1$ converges in polynomial time to C', where*

$$(\alpha\text{-}core(C) - S_{\geq\alpha}) \subseteq (C' - S_{\geq\alpha}) \subseteq (\alpha\text{-}neighborhood(C) - S_{\geq\alpha})$$

with probability at least $(1 - \delta)$, where $S_{\geq\alpha}$ is a set with probability less than γ.

Proof 3.1 Let $0 < \alpha, \delta, \gamma < 1$. Lemma 3.2 states that, if $N > \frac{\lceil\sqrt{2}\rceil^2}{\alpha}\frac{1}{\gamma}\ln\frac{\lceil\sqrt{2}\rceil^2}{\alpha}\frac{1}{\delta}$, then any N randomly-selected training instances will form an $\langle\alpha, \gamma\rangle$-net (with probability at least $(1-\delta)$ for C. Let $S_{\geq\alpha}$ be the set of instances in [0–1,0–1] that are not within α of one of the N instances in the training set.

By definition, C' is the set of instances that the IB1 algorithm predicts will belong to C. More precisely, C' is the set of instances whose nearest (i.e., most similar) training instance is in C.

Two inclusions need to be proven. The first inclusion to prove is to show that, excluding the instances of $S_{\geq\alpha}$, the α-core of C is contained in C' (and thus in $C' - S_{\geq\alpha}$). Let p be an arbitrary instance in the α-core of C not in $S_{\geq\alpha}$ and let s be its nearest training instance.

Since the distance between s and p is less than α and p is in the α-core, then s is also in C. Thus s correctly predicts that p is a member of C. Equivalently, this shows that p is a member of C'. Consequently, (α-core $(C) - S_{\geq\alpha}) \subseteq (C' - S_{\geq\alpha})$.

The second inclusion states that $C' - S_{\geq\alpha}$ is contained in the α-neighborhood of C. This can be proven by showing that, if p is outside the α-neighborhood of C, then p is outside of $C' - S_{\geq\alpha}$. Let p be an arbitrary instance outside the α-neighborhood of C and let s be its most similar neighbor. If p is not in $S_{\geq\alpha}$, then s is within α of p, so s is outside of C. In this case, s correctly predicts that p is not a member of C. Since no instance outside the α-neighborhood of C, excluding instances in $S_{\geq\alpha}$, is predicted by C' to be a member of C, then $(C' - S_{\geq\alpha}) \subseteq (\alpha\text{-neighborhood}(C) - S_{\geq\alpha})$. ■

Pictorially, this proof guarantees that, given a confidence, if a sufficient number of training instances are available (polynomially bounded), then the approximation C' of C is contained in the α-neighborhood of C and contains the α-core of C, ignoring a set of probability less than γ. Thus there are exactly two situations that could occur that lead to prediction errors (i.e., where C' could falsely classify p): (1) when $p \in \alpha$-neighborhood of C and $p \notin \alpha$-core of C or (2) when $p \in S_{\geq\alpha}$.

The following theorem extends this result to $\Re^d$ and $k \geq 1$.

Theorem 3.2 *Let C be any region bounded by a closed curve in a bounded subset of $\Re^d$. Given $0 < \alpha, \delta, \gamma < 1$, then the IB1 algorithm with $k \geq 1$ converges in polynomial time to C', where*

$$(\alpha\text{-core}(C) - S_{\geq\alpha}) \subseteq (C' - S_{\geq\alpha}) \subseteq (\alpha\text{-neighborhood}(C) - S_{\geq\alpha})$$

with probability at least $(1 - \delta)$, where $S_{\geq\alpha}$ is a set with probability less than γ.

Proof 3.2 Let $0 < \alpha, \delta, \gamma < 1$. Lemma 3.4 states that, if

$$N > \frac{k\lceil\frac{\sqrt{d}}{\alpha}\rceil^d}{\gamma} \ln \frac{\lceil\frac{\sqrt{d}}{\alpha}\rceil^d}{1 - \sqrt[k]{1-\delta}},$$

then any N randomly-selected training instances will form a k-$\langle\alpha, \gamma\rangle$-net for C (with probability at least $(1 - \delta)$). Let $S_{\geq\alpha}$ be the set of instances in a bounded subset of $\Re^d$ that are not within α of k of the N training instances.

By definition, C' is the set of instances that the IB1 algorithm predicts will belong to C. More precisely, C' is the set of instances whose k nearest (i.e., k most similar) training instances are in C.

Two inclusions need to be proven. The first inclusion to prove is to show that, excluding the instances of $S_{\geq\alpha}$, the α-core of C is contained in C' (and thus in $C' - S_{\geq\alpha}$). Let p

be an arbitrary instance in the α-core of C not in $S_{\geq\alpha}$ and let K be its set of k nearest (most similar) training instances. Since the distance between each $s \in K$ and p is less than α and p is in the α-core, then each s is also in C. Thus K correctly predicts that p is a member of C. Equivalently, this shows that p is a member of C'. Consequently, (α-core $(C) - S_{\geq\alpha}) \subseteq (C' - S_{\geq\alpha})$.

The second inclusion states that $C' - S_{\geq\alpha}$ is contained in the α-neighborhood of C. This can be proven by showing that, if p is outside the α-neighborhood of C, then p is outside of $C' - S_{\geq\alpha}$. Let p be an arbitrary instance outside the α-neighborhood of C and let K be its set of k most similar neighbors. If p is not in $S_{\geq\alpha}$, then each $s \in K$ is within α of p, so each s is outside of C. In this case, K correctly predicts that p is not a member of C. Since no instance outside the α-neighborhood of C, excluding instances in $S_{\geq\alpha}$, is predicted by C' to be a member of C, then $(C' - S_{\geq\alpha}) \subseteq (\alpha\text{-neighborhood}(C) - S_{\geq\alpha})$. ■

Notice that Theorem 3.2 does not specify what the probability is of the set on which C' and C differ. Rather, it only shows where and how prediction errors could occur. Some constraints on both the length of the boundary of C and the probability of regions of a given area are needed to bound this probability of error.

The following theorem adds these constraints and, in doing so, defines ϵ, the total probability of prediction error, in terms of α and γ. It makes the arbitrary assumption that the amount of prediction error in the two cases described above are equal (i.e., (1) when $p \in \alpha$-neighborhood of C and $p \notin \alpha$-core of C or (2) when $p \in S_{\geq\alpha}$). The proof shows that, for a large class of probability distributions, the IB1 algorithm will, with high probability (i.e., at least $(1 - \delta)$), converge to an approximately correct definition of the target concept (i.e., a $\langle\alpha, \gamma\rangle$-approximation of it) for a large class of concepts in a bounded subset of $\Re^d$ for $d \geq 1$, except for a set with a small probability (i.e., less than γ).

Theorem 3.3 *Let $\mathcal{C}$ be the class of all concepts in a bounded subset of $\Re^d$ that consist of a finite set of regions bounded by closed hyper-curves of total area less than L. Let $\mathcal{P}$ be the class of probability distributions representable by probability density functions bounded from above by B. Then $\mathcal{C}$ is polynomially learnable from examples with respect to $\mathcal{P}$ using IB1.*

Proof 3.3 In Theorem 3.2, if the area of the boundary of C is less than L, then the total area between the α-core and the α-neighborhood of C is less than $2L\alpha$. Then $2LB\alpha$ is an upper bound on the probability of that area. The total error made by C' in Theorem 3.2 is less than $2LB\alpha + \gamma$. If we fix $\gamma = 2LB\alpha = \frac{\epsilon}{2}$, then $\alpha = \frac{\epsilon}{4LB}$ and the theorem follows by substituting these expressions for γ and α into the inequality derived in Lemma 3.2. This yields

$$N > \frac{k\lceil\frac{\sqrt{d}}{\alpha}\rceil^d}{\gamma} \ln \frac{\lceil\frac{\sqrt{d}}{\alpha}\rceil^d}{1 - \sqrt[k]{1-\delta}}$$

$$N > \frac{2k\lceil\frac{4LB\sqrt{d}}{\epsilon}\rceil^d}{\epsilon} \ln \frac{\lceil\frac{4LB\sqrt{d}}{\epsilon}\rceil^d}{1 - \sqrt[k]{1-\delta}}$$ ■

This proof showed that the number of instances required by IB1 to learn this class of concepts is also polynomial in L and B. This suggests that IB1 will perform best when the target concept's boundary size is minimized.

These proofs imply several additional qualitatively important statements.

1. If the α-core is empty, then C' could be any subset of the α-neighborhood of C. (C's α-core is empty when all elements of C are within α of C's boundary, as could occur when C's shape is extremely thin and α is chosen to be too large.) The IBL approximation of C could be poor in this case.

2. IB1 cannot distinguish a target concept from anything containing the α-core and contained in its α-neighborhood. Consequently, small perturbations in the shape of a target concept are not captured by this approach. However, only near-boundary instances will provide IB1 with information concerning where the concept boundary lies. Instances far from the concept boundary can be ignored without damaging the accuracy of the approximation. This is the thesis of IB2, discussed in Section 4.2, which reduces storage requirements without greatly sacrificing classification accuracy by storing only near-boundary instances.

3. Except for a set of size less than γ, the set of false positives is contained in the outer ribbon (the α-neighborhood of C excluding C) and the set of false negatives is contained in the inner ribbon. As we will see, IB2 works by saving only those instances for which prediction errors occur. These instances approximate the set of near-boundary instances in the training set.

4. Noisy training instances will fool IB1 into thinking that part of the concept boundary lies nearby the noisy instance. If such instances are distant from the actual concept boundary, then they will be misclassified and stored by IB2. This dramatically increases its storage requirements and reduces its predictive accuracy (i.e., the stored noisy instance can misclassify subsequently presented and nearby noise-free training instances. Most noisy instances are distinguished by their poor predictive accuracy on subsequently presented training instances. IB3, an extension of IB2 described in Section 4.3, uses this information to detect and remove noisy instances from partial concept descriptions.

5. As the dimensionality of a target concept increases, the expected number of instances required to learn (closely approximate) it will increase exponentially. In particular, IB1 will require exponentially more training instances to closely approximate a concept with linear increases in the dimensionality of the instance space. Space transformation methods that reduce this dimensionality will significantly increase the efficiency of IBL algorithms. This is the basis of IB4, which will be described in Section 4.4.

6. No assumptions about the convexity of the target concept, its connectedness (number of components or disjuncts), nor the relative positions of the various components of the target concepts need be made.

The primary conclusion of these proofs is that $\mathcal{C}$ is polynomially learnable by the IB1 algorithm if both the α-core and α-neighborhood of C are good approximations of C.

3.3.2 Convergence Theorems: Predicting Numeric Values

The PAC-learning model required only a small extension to show which class of symbolic-valued concepts are polynomially learnable by IB1. A more radical extension of the model is required to show which class of *numeric functions* is polynomially learnable. This section introduces the notion of PAC-learnability for predicting numeric values and proves that IB1, given a sufficiently-sized training set drawn according to any arbitrary fixed distribution, can learn the class of continuous functions with bounded slope. Theorem 3.4 establishes this for the situation when $k \geq 1$ and the dimensionality d of the instance space is 1. Theorem 3.5 extends this to the case when $d \geq 1$.

A few definitions are needed for the theorems.

Definition 3.11 The *error of a real-valued function* f' in predicting a real-valued function f, for an instance x, is $|f(x) - f'(x)|$.

Definition 3.12 Let f be a real-valued target function. Let B_f be the least upper bound of the absolute value of the slope between any two instances on the curve of f. If B_f is finite then we say that f has *bounded slope.* If $\mathcal{C}$ is a class of functions in which each $f \in \mathcal{C}$ has bounded slope, then we say that $\mathcal{C}$ has bounded slope.

As an example, the class of continuously differentiable functions on $[0, 1]$ has bounded slope. As a counterexample, the function $f(x) = \sin(\frac{1}{x})$ does not have bounded slope.

Definition 3.13 describes the extension of the PAC-learning model assumed by the theorems in this section.

Definition 3.13 Let $\mathcal{C}$ be a class of functions for which each $f \in \mathcal{C}$ is a function with bounded slope from the unit hyper-cube in $\Re^d$ to $\Re$. $\mathcal{C}$ is *polynomially learnable* from examples if there exists an algorithm A and a polynomial p such that, for any $0 \geq \alpha, \alpha, \delta < 1$ and $B > 0$, given an $f \in \mathcal{C}$ with $B_f \leq B$, if more than $p(\frac{1}{\gamma}, \frac{1}{\alpha}, \frac{1}{\delta}, B)$ examples are chosen according to any fixed probability distribution on $[0, 1]$. then, with confidence at least $(1-\delta)$, A will output an approximation of f with error less than α (except on a set of instances with probability less than γ).

The variable γ denotes the same value as it did for predicting symbolic values (i.e., the probability of the set to be ignored). However, this definition seems unusual because ϵ is missing. In this case, the set on which prediction "errors" occur is γ. A prediction error is now defined as a prediction that differs by more than α from the actual target value.

IB1's method for predicting numeric values was first described in Section 2.3. Given a good set of training instances, IB1 can generate a piecewise linear function that is a good approximation to an unknown continuous function. It calculates an instance's similarity to all instances in a concept description, selects the k most similar instances, and predicts that the target value is the similarity-weighted prediction derived from those instances.

The following theorem demonstrates that continuous, real-valued functions with bounded slope in the unit interval $[0, 1]$ are polynomially learnable by IB1. Note that the class of functions with bounded slope includes the large class of continuously differentiable functions.

Theorem 3.4 *Let C be the class of continuous, real-valued functions on $[0, 1]$ with bounded slope. Then C is polynomially learnable by IB1 with $k \geq 1$.*

Proof 3.4 Let f be a continuous function on $[0, 1]$. Let α, γ, and δ be arbitrary positive numbers less than 1. The bound B_f will be used to ensure that function f is guaranteed to not vary much on a small interval.

The proof follows after assuming the training set forms a k-$\langle \alpha, \gamma \rangle$-net. Lemma 4.4 guarantees that, if $d = 1$ and if

$$N > k \frac{1}{\alpha' \gamma} \ln \frac{1}{\alpha'(1 - \sqrt[k]{1 - \delta})},$$

then, with confidence at least $(1 - \delta)$, every instance of $[0, 1]$ will be within α' of at least k of the selected N training instances, except on a set $S_{\geq \alpha}$ of probability less than γ.

Let B_f be an upper bound on the slope of f, let f' be the approximation (i.e., concept description) that IB1 generates for f, and let instance x be an arbitrary instance in $S_{<\alpha}$, the set of instances not in $S_{\geq \alpha}$. The point is to ensure that the error of f', for every instance $x \in S_{<\alpha}$, is small (i.e., less than α). That is, it must be shown that $|f(x) - f'(x)| < \alpha$. The proof will solve for α' in terms of B_f and α.

IB1's approximation f' predicts that the target value of an instance x is the similarity-weighted average of its k most similar training instance's target values. Let K be the set of x's k most similar training instances. Since $f'(x)$ is a weighted-average of the values of the target values of each $x' \in K$, it suffices to show that $|f(x) - f(x')| < \alpha$. If this can be proved, then we are guaranteed that $|f(x) - f'(x)| < \alpha$.

Because N is sufficiently large, the k most similar neighbors of x must all be within α'

of x. Also, we know that B_f is the upper bound on the slope between x's target value and the target value of x'. Thus, since

$$|f(x) - f(x')| = \text{slope}(x, x') \times \text{distance}(x, x'),$$

then

$$|f(x) - f(x')| < B_f \times \alpha' = \alpha.$$

Solving for α', we find that $\alpha' = \alpha / B_f$. Therefore, IB1 will yield a prediction for x's target value that is within α of x's actual target value if at least

$$N > \frac{kB_f}{\alpha\gamma} \ln \frac{B_f}{\alpha(1 - \sqrt[k]{1-\delta})}$$

training instances are provided. ■

Notice that the above proof also shows that, for the class of continuous functions with slope bounded by a single constant B, the number of examples required (and consequently the time required) to learn any concept in the class is dependent only on the level of confidence and accuracy desired, and is independent of the particular element of the class to be learned. This permits IB1 to learn concepts in the class, and know when it is safe to stop, without knowing the maximum value of the slope for the particular concept being learned. On the other hand, it requires that the value of a bound on the slope be known for the entire class of target concepts.

This theorem indicates that, given an open domain, some constraints are required on the "wildness" of the function to ensure that the time to learn is polynomially bounded. In particular, looser constraints on the domain of the function are compensated for by tightening the constraints on how fast the function can change its value over a small interval.

The following theorem extends Theorem 3.4 to instance spaces defined in $\Re^d$.

Theorem 3.5 *Let C be the class of continuous, real-valued functions on a bounded subset of $\Re^d$ with bounded slope. Then C is polynomially learnable by IB1 with $k \geq 1$.*

Proof 3.5 This theorem is identical to Theorem 3.4 except that the dimensionality is $d \geq 1$ rather than $d = 1$. The proofs are almost identical. As before, the slope of the target function f is assumed to be bounded by B_f. It can be shown that, if

$$N > \frac{k\lceil \frac{B_f\sqrt{d}}{\alpha} \rceil^d}{\gamma} \ln \frac{\lceil \frac{B_f\sqrt{d}}{\alpha} \rceil^d}{1 - \sqrt[k]{1-\delta}}$$

training instances are sampled, then, with confidence at least $(1 - \delta)$, $|f(x) - f'(x)| < \alpha$ is true for all instances x in the bounded subset of the instance space $\Re^d$ (except for a set of instances with probability less than γ). ■

Many functions have bounded slope. For example, any piecewise linear curve and any function with a continuous derivative has a bounded slope. Therefore, IB1 can accurately learn a large class of numeric functions in polynomial time. However, it cannot PAC-learn numeric functions whose maximum absolute slope is unbounded. For example, IB1 cannot learn $f(x) = \sin(\frac{1}{x})$ in polynomial time. As x approaches 0, the derivative (slope) of this function is unbounded. For every $0 < \alpha < 1$, there is no piecewise linear α-approximation of this function.

3.4 Analyzing Other IBL Algorithms

The preceding section is severely limited: the only IBL algorithm analyzed was IB1. This algorithm has several critical limitations that constrains the set of real-world applications for which it would be the most appropriate IBL algorithm. For example, it has high storage requirements. The theorems show that IB1's storage requirements increase exponentially with the dimensionality of the instance space. Therefore, IB1 would be a poor choice for applications involving multiple irrelevant attribute dimensions.

The algorithms introduced in Chapter 4 are extensions of IB1 that improve IB1's learning behavior. However, they are far more complex than IB1. Consequently, they are not as amenable to analysis and no proofs of convergence have been constructed for them. Their complexities concern issues such as the presentation ordering of training instances, instance-averaging, instance accuracy information, and attribute weights. These algorithms will be empirically investigated in Chapter 4 rather than mathematically analyzed here.

It is useful to briefly examine one of these issues to demonstrate why these algorithms are more difficult to analyze. The theorems described in this chapter gave a good characterization of the classes of concepts learnable by an *instance-saving* IBL algorithm (IB1). The situation for *instance-averaging* IBL algorithms is more complex. Bradshaw (1987), Sebestyen (1962), Kohonen (1986), and several others have demonstrated that instance-averaging algorithms work in real domains. However, it is difficult to find any reasonable constraint on the concept shape that would guarantee convergence for these algorithms.

For example, if the target concept was shaped like a ring and the learning algorithm was given only positive examples, then instance-averaging approaches would converge to the center of the mass (centroid) of the ring. Thus they would converge to an instance which was not even a member of the concept! Although this is an extreme example, there is always the possibility that an instance-averaging algorithm will converge to an instance that is not in the concept when its shape is not convex.

A reasonable constraint might be that instance-averaging algorithms will converge if the concept is convex. However, even this strong constraint is not sufficient. Consider

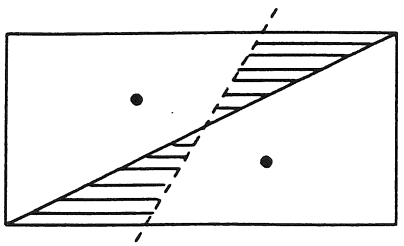


Figure 3.3: An instance-averaging example. The darkened regions denote the sets of instances that would be misclassified.

the rectangular instance space in Figure 3.3. The boundary between positive and negative instances in this instance space is the diagonal delineated by the solid line. If the training set is ordered so that all positive instances precede all negative instances, then instance-averaging algorithms will converge to the centroids (shown in the figure). The shaded area would then represent the error set for instance-averaging.

It is not obvious how to prove which classes of concepts are polynomially learnable by IBL algorithms that perform instance-averaging. Empirical investigations (Chapter 4) have not yet confirmed that they are superior to instance-saving learning algorithms. Nor have they been shown to perform worse then other IBL algorithms in measures of classification accuracy and storage requirements. However, instance-saving algorithms seem to be more attractive because they are easier to analyze, are simpler, and have performed as well.

3.5 Chapter Summary

This chapter addressed the performance dimension of generality. Several theorems were detailed. They showed that IB1 can learn, in polynomial time, a large class of concepts: (1) the class of concepts describable as a finite union of regions with finite boundary length and (2) the class of continuous functions with bounded slope. Extensions of the Valiant (1984) PAC-learning model were introduced to allow for the analysis of IBL algorithms with respect to a given class of probability distributions. The theorems described in this chapter are unusual for PAC-learning analyses in that the instance space consists only of numeric-valued attribute dimensions, the target concept being learned could be a numeric-valued function, and the focus was on a particular algorithm (IB1). Furthermore, the proofs relied heavily on geometric interpretations of the instance space. However, they have been presented in a form that preserves the central focus of the PAC-learning model (i.e., learnability in polynomial time).

Convergence theorems have only been provided for IB1, and only for numeric-valued predictor attribute dimensions. IB1 is far more simple to analyze than are the other IBL algorithms described in this dissertation. Empirical analyses will be described in Chapter 4 for investigations involving symbolic-valued attributes and more elaborate IBL algorithms. Although it is possible to show that the class of concepts learned by these algorithms is PAC-learnable (i.e., by some algorithm), it is difficult to generate these proofs for the specific algorithms detailed in Chapter 4 (Volper & Shackelford, 1987).

The analyses presented in this chapter suggest how storage requirements could be decreased and how irrelevant attributes could be tolerated. These are some of the topics that are investigated in the next chapter, which details empirical analyses of IBL algorithms that are extensions of IB1.

Chapter Acknowledgements

Marc Albert and Dennis Kibler were primarily responsible for the development of the proofs described in this chapter. Dennis Volper also provided several key insights concerning these analyses. Thanks to Marc and Wendy Sarrett for reading and providing comments on earlier drafts of this chapter.

Chapter 4
Empirical Analyses

Murphy's corollary: if you can analyze it mathematically, the algorithm is far too slow and space consumptive to work in practice.
–Dennis Volper

The significant contribution of this chapter is a demonstration that the IBL paradigm specifies algorithms for efficiently solving difficult problems associated with supervised learning tasks. In this chapter I empirically examine a comprehensive sequence of three extensions to IB1. Each extension addresses a specific limitation inherent in IB1 that was anticipated by the mathematical analysis in Chapter 3. In this sense, our mathematical analyses directed our empirical analyses of IBL algorithms.

The three algorithms that will be studied in this chapter are named *IB2*, *IB3*, and *IB4*. IB2 (Section 4.2) addresses the problem of excessive storage requirements. Our PAC-learning analyses suggested that only near-boundary instances need to be saved to distinguish where a concept's boundary lies when IB1 is used to learn symbolic-valued concept descriptions. Similarly, when predicting numeric values, IB1 needs to save only those instances that lie in areas of the instance space where the target function has a sufficiently high absolute slope. More generally, the most informative instances are those that lie near areas where the derivative of the instance space with respect to the target values is high. Given only these instances, IB1 would still generate a reasonably accurate approximations. IB2 attempts to save only these instances in its partial concept descriptions.

However, IB2 can easily be fooled. IB2 is highly sensitive to noisy instances, which usually tend to resemble boundary instances (or instances in areas of high absolute slope). Moreover, this problem isn't limited to applications with noisy instances. Noise-free instances can sometimes resemble noisy instances when the attributes used to describe them cannot perfectly distinguish concept members from non-members. IB3 (Section 4.3) is an extension of IB2 that corrects this problem. It allows only instances that perform well (i.e., that have recorded significantly good predictive accuracies on subsequently presented training instances) to assist in generating classification decisions.

IB3 is also flawed. Like its predecessors, it assumes that the predictor attributes are equally relevant for predicting target values. This is often an incorrect assumption. IB4

(Section 4.4) is an extension of IB3 that learns the relative relevance of attributes for predicting a target attribute's values. This significantly speeds up IB3's learning rate, which, as the mathematical analysis hinted, decreases exponentially with linear increases in the number of predictor attributes.

These three algorithms are difficult to examine mathematically due to their complexity. Furthermore, mathematical analyses, which are usually restricted to simple artificial domains, are not especially appropriate for the goals of this chapter, which are to show that, the IBL paradigm specifies algorithms that can perform as efficiently as other established algorithms in practical supervised learning applications. This chapter begins with a discussion in Section 4.1 concerning the scope and experimental methodology used in the experiments in this chapter. These experiments investigate claims for IB2, IB3, and IB4's learning behavior. Sections 4.2 through 4.4 summarize evidence of these algorithms' ability to solve their intended tasks. Finally, Section 4.5 summarizes some of the problems with IB4.

Noticeably absent from this chapter are *parametric studies*, which are evaluations of the sensitivity of design choices made by algorithm designers. For example, only one setting for k is used in all the experiments described in this chapter (i.e., $k = 1$). Instead, this chapter focuses on task-specific solutions, simply showing that some IBL algorithms exist for solving well-known problems encountered in supervised learning tasks. Parametric studies are postponed until Chapter 5.

4.1 Scope and Experimental Methodology

IB1 is identical to the k-nearest neighbor algorithm except that it normalizes the ranges of the numeric-valued predictor attributes, employs a weighted-similarity prediction algorithm for numeric prediction tasks, and processes instances incrementally rather than in batch. Domain-independent IBL algorithms like IB1 suffer from several problems that must be solved before they can be expected to perform well in practical learning applications. Breiman, Friedman, Olshen, and Stone (1984) wrote what is probably the most comprehensive critique of the nearest neighbor algorithm published by practitioners interested in learning algorithms. Their critique, taken both from the list on page 17 and from statements throughout their book, included the following observations:

1. they are computationally expensive classifiers because they save and compute similarities to all training instances,
2. they are intolerant of noise,
3. they are intolerant of irrelevant attributes,
4. they are sensitive to the choice of the algorithm's similarity function,
5. there is no natural or simple way that they can use to tolerate missing attribute values,
6. there is no natural or simple way they can use to process symbolic-valued attribute values, and
7. they give little usable information regarding the structure of the data.

These claims are probably sufficient to frighten most users from considering using a learning algorithm based on the nearest neighbor approach! However, Breiman and his colleagues didn't investigate potential solutions to these problems with the basic algorithm. Several of these problems with the basic approach are at least partially solvable through suitable extensions to the nearest neighbor algorithm.

The algorithms introduced in this chapter are extensions of IB1 that have different definitions for their framework components. Therefore, they can be viewed as improvements to the k-nearest neighbor algorithm ($k \geq 1$). More specifically, IB2, IB3, and IB4 represent *efficiency improvements* to IB1 measured in terms of reduced storage requirements and/or increased learning rate, where this rate is defined as the number of instances that are required to attain a given classification accuracy on a set of test instances. IB2 addresses the first problem listed above. It is more storage efficient than IB1. IB3 addresses the second problem. It is both more storage efficient than IB2 and has a faster learning rate than both IB1 and IB2 when noise exists in the training set. Finally, IB4 addresses problem numbers 3 and 4. It tolerates irrelevant attributes by tailoring its similarity function to the application domain. This allows IB4 to have a faster learning rate than its predecessor IBL algorithms when the attributes used to describe instances vary greatly in their relevance for predicting target values.

Progress has also been made on the remaining problems listed above. I described a simple method for tolerating missing attribute values in Section 2.3. However, it is a primitive algorithm that conveniently ignores deeper issues involved with missing values. Section 5.2.1 presents an improved method for tolerating missing attribute values. Section 2.3 also described a trivial algorithm for defining similarity over symbolic-valued attributes. However, Stanfill and Waltz (1986), among others, have already demonstrated that there exist similarity functions defined for symbolic-valued attributes which support efficient learning behavior. Cost and Salzberg (1990) demonstrated that this similarity function can work well on other applications. Finally, Salzberg (1990) described an IBL algorithm that builds hyperrectangular partitions of the instance space. These partitions, once ordered, resemble the same partitions learned by decision tree algorithms. Many researchers have argued that decision trees yield comprehensible concept descriptions that reveal the structure of the

data (Breiman *et al.*, 1984; Quinlan, 1986a; Michie, 1990). It appears that extended IBL algorithms can also yield concise descriptions of concepts.

Some of the empirical analyses described in this chapter will use artificial domains, which allow for the systematic variation of the settings for independent variables. These analyses enable studies of the affect that an independent variable's setting has on a dependent variable (e.g., classification accuracy as the probability of noise-corrupted training instances is varied). Although it is possible to carry out these studies using databases with unknown target concepts (Quinlan, 1986b), the experimenter runs the risk of not knowing the specific interdependencies among the characteristics that define the application domain. For example, if some of the attributes describing the database's instances are unknowingly irrelevant, then evaluations of an algorithm's sensitivity to noise added to these attributes can lead to incorrect conclusions. However, results from studies with artificial domains do not necessarily reflect how algorithms will perform on databases whose instances were derived from real-world observations. Therefore, empirical analyses with databases containing collected sets of instances pertaining to real-world prediction tasks will also be described. These studies will be used to determine whether certain domain characteristics that are typical of real-world databases have been overlooked. Comparisons with other inductive learning algorithms will also be used to evaluate the learning behavior of the IBL algorithms.

The dependent variables examined in the empirical analyses described in this chapter include learning rate, storage requirements, predictive accuracy (i.e., classification accuracy for symbolic prediction tasks and average relative error for numeric prediction tasks), and the quality of the instances in the concept description, defined as the percentage of instances used in classification predictions that are not noisy. Independent variables will include the amount of training, the probability of noise among training instances, the number of irrelevant attributes used to describe instances, the learning algorithm, and the application domain.

Several dimensions of potential empirical analysis are ignored in this chapter. This allows the discussion to focus on the main problem addressed by algorithms IB2, IB3, and IB4 respectively. The ignored dimensions include, among others, the algorithm used to tolerate missing attribute values, the definition of similarity for symbolic-valued attributes, the types of attributes used to describe instances, and the value of k (the number of most similar instances used to determine predictions). The settings for these dimensions will be held fixed in the experiments in this chapter rather than varied as independent variables. However, several of these variables will be investigated in Chapter 5.

4.2 IB2: Reducing Storage Requirements

Breiman and his colleagues are correct; learning algorithms based on variants of the nearest neighbor classification strategy need to address the problem of high storage requirements. A critical reason for reducing storage requirements is to decrease the time required to make predictions, which depend on computing the similarity of a novel instance with all the instances stored in a partial concept description.

4.2.1 Methods for Improving Retrieval Efficiency

One way to alleviate this problem is to use a smart indexing strategy. For example, Moore (1990) used a k-d tree to index instances in his application of the nearest neighbor function to solve robotic manipulation tasks. A k-d tree is a binary tree such that at each depth a different attribute value is tested to determine which branch is to be followed. Moore reported $O(\log n)$ time requirements for an algorithm that locates an instance's nearest observed instance in a k-d tree. However, this level of performance is an artifact of Moore's specific applications and cannot be duplicated in general. Sproull (in press) showed that the number of instances in the tree must grow exponentially with k to preserve logarithmic search times. Sproull's empirical studies showed that, although logarithmic behavior is attainable for all but tiny trees, extremely large numbers of instances are required (i.e., for $k = 16$, a search of a 76,000-instance k-d tree examines almost every instance). Not surprisingly, Moore's k-d trees displayed logarithmic retrieval behavior because his applications involved using thousands of instances with low dimensionality (i.e., a low value for k).

Moreover, this does not reduce overall storage requirements. Moore postulated that the size of the trees produced could be reduced by an order of magnitude if only a subset of the instances were saved. I agree; perhaps IB2 (detailed below) could be used to accomplish this task.

Several researchers have demonstrated that, without some means of restraint, learning algorithms tend to *overfit* their application domain (Sturt, 1981; Breiman *et al.*, 1984; Quinlan, 1986a; Michalski *et al.*, 1986; Clark & Niblett, 1989). Overfitting occurs when the concept description output by a learning algorithm relies too heavily on the assumption that the training data completely accounts for the idiosyncrasies inherent in the application domain. Subsequently, several experiments showed that higher classification accuracies can be obtained by post-pruning lower nodes from decision trees and truncating rules that account for few instances. The removed parts of the concept description did not assist in maximizing predictive accuracy on test instances. Similar storage techniques can be used in IBL algorithms, which is the topic of Section section:ib2-description.

4.2.2 Description of the IB2 Algorithm

The mathematical analyses in Chapter 3 showed that, although instance-based learning algorithms have high storage requirements, they do not always need to save all observed training instances to maximize predictive accuracy. In particular, only those instances nearby the concept boundary need be saved for this purpose when predicting symbolic values. For numeric prediction tasks, those instances located in areas of the space where the target value changes rapidly (i.e., where the second derivative with respect to the target value is high) should be saved. Instances that are distant from the concept boundary do not contain information required to obtain accurate predictions because they do not delineate where the concept boundary lies. This is also true for instances that are in areas where the target function has ; accurate numeric predictions can be obtained without reference to these instances.

However, IBL algorithms are not told where the concept boundary lies or where in the instance space the target function varies greatly. (If this information was available, then the problem, that of supervised concept learning, would be solved.) One way to estimate the amount of prediction-related information contained in an instance is to evaluate the degree to which the predictions made by the concept description change when an instance is included in it. Since IBL algorithms do not maintain abstractions, this approach is expensive; it requires testing many instance's classification before and after the inclusion of the new instance to estimate the amount of change in the algorithm's predictive behavior.

An alternative method for reducing storage requirements is based more closely on the mathematical analysis, which showed that, once given enough instances, prediction errors are constrained to occur only nearby the concept boundary or in areas where the numeric target value changes rapidly. An IBL storage reducing algorithm could possibly wait until it has saved sufficient numbers of instances and then discard those that would otherwise be classified correctly (Kibler & Aha, 1987). However, this approach is unsatisfactory since it saves many instances before deleting them. Instead, this behavior can be approximated by incrementally retaining only those instances for which predictions are inaccurate.

This behavior describes IB2's training algorithm, which is detailed in Table 4.1.[1] For symbolic prediction tasks, it saves only misclassified instances (i.e., the tolerance_threshold is set to 0). For numeric prediction tasks, it saves only those instances whose target value prediction error is above a parameterized tolerance threshold. This allows IB2 to save a set of instances that closely approximates those that should be saved.

In summary, IB2's training algorithm is identical to IB1's except that it saves only a subset of the training instances. Its testing algorithm is identical to IB1's, shown in Table 2.2

[1]See Table 2.1 on page 21 and Section 2.3 for a description of IB2's sub-functions that are not defined in Table 4.1.

Table 4.1: The IB2 training algorithm.

```
Key: T: Training set
     P: Set of predictor attributes
     t: The target attribute
     k: Number of most similar instances used
     PCD: Partial concept description

Train(T, P, t, k,tolerance_threshold)
   1. Set global variables
   2. for each x_i ∈ T do
      2.1 x_i ← Pre_process(x_i, P)
      2.2 prediction ← Performance(x_i, P, t, k)
      2.3 Learn(x_i,prediction,t,tolerance_threshold)

Learn(x,prediction,t,tolerance_threshold)
   if (|x_t − prediction| > tolerance_threshold)
      then PCD ← PCD ∪ {x}
```

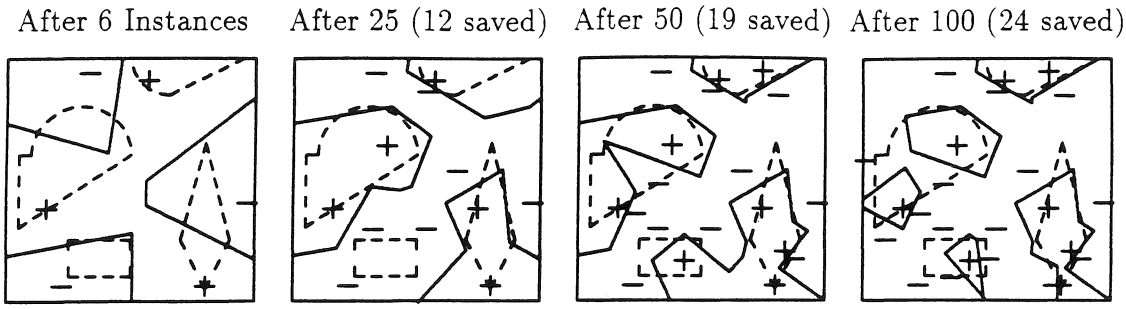
on page 23. The following sections examine IB2 when $k = 1$. Section 5.2.2 summarizes IB2's behavior when $k \geq 1$.

4.2.3 An Examination of IB2's Learning Behavior

Symbolic Prediction Tasks

Figure 4.1 summarizes IB2's behavior on the training set given previously to IB1 (See Figures 2.3 and 2.4 on pages 27 and 27). At first, IB2's behavior is identical to IB1's because they both save the first six instances, which were misclassified by IB2. IB2's accuracy is higher than IB1's after processing the first 25 and first 50 training instances even though it saved fewer instances. Although IB2 saved only 24 training instances, its accuracy after processing 100 training instances is almost as good as IB1's. Its accuracy would have been as good as IB1's had it saved only a few more near-boundary instances.

Although the application shown in Figure 4.1 demonstrates how IB2's accuracy can be nearly as good as IB1's with only a fraction of the storage requirements, it is difficult to see that IB2's saved instances were primarily near-boundary instances. It is more obvious that, after the first few training instances have been presented, IB2 primarily saved only near-boundary instances in the application summarized by Figure 4.2, where the two-dimensional target concept has only one disjunct. A clear majority of the misclassified instances lie inside



IB2's Accuracy: 51.1%	74.1%	79.9%	86.0%
IB1's Accuracy: 51.1%	71.3%	83.8%	88.4%

Figure 4.1: IB2's saved instances and its predicted concept boundaries after 4 points during the training process. In this case, the value for k was set to 1.

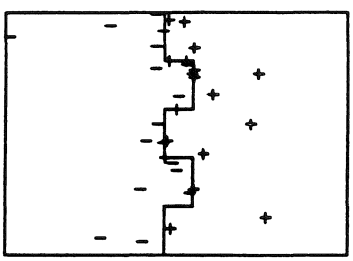


Figure 4.2: The majority of IB2's saved instances are close to the concept's boundary (denoted by the jagged solid line). Concept members and non-members are labeled with "+" and "-" respectively.

the α-neighborhood and outside the α-core of the target concept, where α's value decreases with training. These observations lead to an answer for Hart's (1968) question concerning CNN: what are its expected storage requirements?[2] The expected storage requirements for CNN, which also saved instances primarily along concept boundaries, are polynomial in the size of the concept boundary.

IB2 was evaluated on a set of seven carefully chosen applications with diverse instance space characterizations.[3] These application domains, which are available from the University of California at Irvine's repository of machine learning databases and domain theories (send requests to ml-repository@ics.uci.edu), are briefly summarized in Table 4.2. The following

[2]As described in Chapter 7, CNN is identical to IB2 except that it cycles repeatedly through the training set and doesn't normalize its instances.

[3]However, the majority of the attributes describing instances in these applications are Boolean- or numeric-valued. Section 5.2.1 describes experiments with symbolic-valued attributes.

Table 4.2: Database characteristics.

Database Name	Training Set Size	Test Set Size	Number of Attributes	Number of Classes
LED-7 Display	200	500	7 Boolean	10
Waveform-21	300	100	21 Continuous	3
Cleveland	212	91	13 Continuous	2
Hungarian	206	88	13 Continuous	2
Voting	305	130	16 Boolean	2
LED-24 Display	1000	200	24 Boolean	10
Waveform-40	300	100	40 Continuous	3

paragraphs briefly summarizes these applications while Appendix A describes them in more detail.

1. The LED-7 display domain is an artificial domain described in (Breiman *et al.*, 1984, pp. 43–49). Each of the ten target concepts refers to a unique decimal digit's Boolean on/off pattern as described by a set of seven light-emitting diodes. This domain was chosen because it contains noise (i.e., each instance's attribute value is corrupted (negated) with probability 10%) and most of its attributes have high attribute relevance. The experiments with this application should distinguish the behaviors of IB2, which I will show is sensitive to noisy data, and IB3, a noise-tolerant extension of IB2 that is examined in Section 4.3.
2. The Waveform-21 domain, (Breiman *et al.*, 1984, pp. 49–55) is another noisy artificial domain. It is valuable for these experiments because its attributes are continuous rather than Boolean-valued. Furthermore, many of its attributes can be disregarded without sacrificing predictive accuracy. Section 4.4 will examine IB4, an extension of IB3 that can tolerate irrelevant attributes. IB4 should perform well on this domain but poorly on the LED-7 domain, where most of the attributes are highly relevant for predicting classifications.
3. The Cleveland database (Detrano, *et al.*, 1989) contains cardiological diagnoses. It is a less contrived database then the LED-7 and Waveform-21 artificial domains. It is also known to contain noise in the sense that the attributes used to describe the diagnoses/instances are insufficient for achieving perfect classification accuracy; Detrano and his colleagues reported classification accuracies of approximately 75% with this database. IB3's behavior should be easily distinguished from IB2's in this application due to its high level of noise. Some of the attributes used in this database are irrelevant; IB4 should perform relatively well here for this reason.
4. The Hungarian database also contains cardiological diagnoses and was obtained from the same source. It is being used here because previous experiments have shown that IB1 and IB2 perform comparatively poorly on this domain due to its large number

(i.e., 781) of missing attribute values. Results with this database should reinforce the conclusions drawn from the experiments with the Cleveland database. Section 5.2.1 describes an investigation concerning the IBL algorithms' sensitivity to missing attribute values.

5. Many of the attributes describing instances in the Congressional Voting database can also be ignored without sacrifice. However, it is unique here in that it contains only small amounts of noise. IB3 is not expected to perform particularly well on this application application because it is designed to expect that instances are noisy, but IB4's benefits should be evident since several of the attributes are essentially irrelevant (i.e., they are redundant).
6. IB4 is even better suited for the LED-25 artificial domain, which contains an additional 17 irrelevant Boolean attributes. One reason that the LED and Waveform domains are included in this study is that Breiman (*et al.*, 1984) published the results from applying CART and the nearest neighbor algorithm to these domains. These results will be used for comparison purposes.
7. Finally, the Waveform-40 artificial domain was included in this study to show that IB4 can outperform the other IBL algorithms in applications involving many irrelevant and continuously-valued attributes. This domain is a variant of Waveform-21 with an additional 19 irrelevant attributes, whose values are normally distributed with a mean of 0 and a variance of 1.

Some of these applications involve finite-sized databases rather than artificial domains from which an infinite number of instances can be generated. In these cases, 70% of the databases' instances were randomly chosen to be used as training instances. The remaining instances were used to test the algorithms' predictions. Although this 70-30 split method has no theoretical basis, it has often been used in empirical comparisons (e.g., Breiman *et al.*, 1984; Clark & Niblett, 1989) and therefore allows for comparisons with results reported for other learning algorithms. Moreover, it is far less expensive to use than a "leave one out" strategy, an extreme form of a cross-validation study, which is an excellent evaluation method, but is practical for only small-sized databases (e.g., Towell, Shavlik, & Noordewier, 1990). The same number of training instances were used for the LED-7 and the two Waveform artificial domains as were used in the CART book (Breiman *et al.*, 1984). A larger number of instances (1000) were used to train the algorithms on the LED-25 artificial domain than used to test CART (Breiman *et al.*, 1984) to show how IB4 learns more quickly than the other IBL algorithms in the presence of numerous irrelevant attributes. The reported results are averages from 25 trials, where different training and test sets were generated for each trial.

Table 4.3 summarizes the results obtained from applying IB1 and IB2 to these data sets. In summary, these results show that IB2 can significantly reduce IB1's storage requirements. However, IB2 sacrifices classification accuracy; its accuracy was consistently lower than IB1's.

Table 4.3: Percent accuracy ± standard error and percent storage requirements.

Application	IB1	IB2
LED-7 display	71.6±0.4 100	63.0±0.9 41.6
Waveform-21	75.5±1.1 100	68.4±1.1 32.3
Cleveland	75.1±0.8 100	71.4±0.9 32.0
Hungarian	56.1±2.2 100	53.1±2.4 36.9
Voting	91.8±0.4 100	90.9±0.5 11.6
LED-24 display	47.9±0.6 100	43.7±0.8 60.1
Waveform-40	68.6±0.7 100	64.0±0.7 38.3

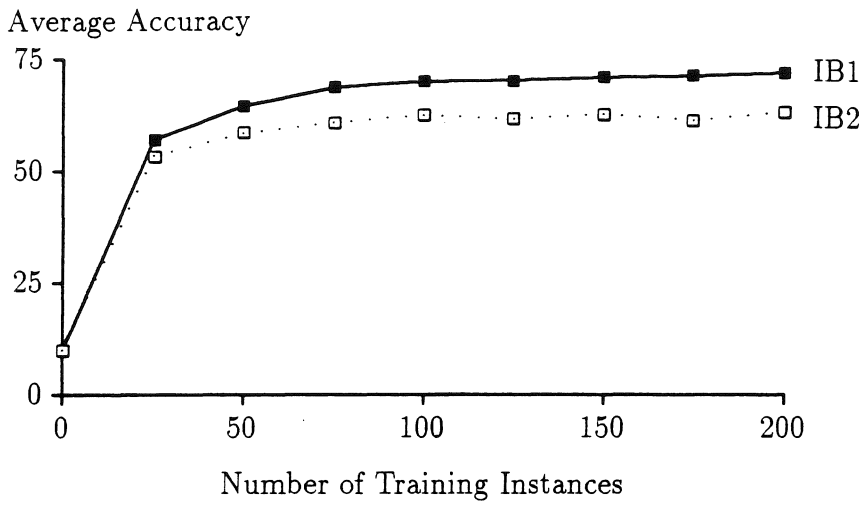


Figure 4.3: IB1's learning curve rises significantly faster than IB2's for the LED-7 application.

For example, Figure 4.3 shows their average learning curves for the LED-7 domain application. IB1's average learning curve rises significantly faster than IB2's ($t(7) = 4.23, p < 0.0025$). More detailed analyses are presented in the following paragraphs.

Noisy Artificial Domains:

The results with the first two artificial domains showed that training set noise strongly degrades IB2's performance. The LED-7 and Waveform-21 artificial domains both contain large amounts but different types of training set noise. The noise added to the first domain logically negates attribute values with a probability of 10%. The noise in the second domain is the addition of a normally distributed error (with mean 0 and variance 1) to each continuously-valued attribute. The value predicted in the LED-7 domain is the digit represented by the seven LED displays. The target value in the Waveform-21 domain is the class of the waveform, which is described by a linear combination of two (of three) waves, where each wave is represented by 21 continuously-valued attributes values. Although IB2

significantly reduced IB1's storage requirements in both applications (i.e., it saved an average of less than 42% of the training instances in both applications), it sacrificed classification accuracy (i.e., at least 7% for each application). This occurred because IB2 misclassified and, subsequently, saved most of the noisy instances. Saved noisy instances were then used, without discretion, to misclassify subsequently presented training instances. This motivates the need for IB3, the noise-tolerant extension of IB2 described in Section 4.3.

Learning Concepts with Imperfect Attribute Sets:

A set of attributes are *imperfect* when they cannot completely distinguish positive from negative instances. The attribute sets for the noisy artificial domains are perfect, but additional noise prevents perfect classification accuracies. However, the attribute sets for the Cleveland and Hungarian databases are imperfect; perfect classification accuracies are unattainable even though no additional noise is added. This occurs because the description language, defined by the set of predictor attributes, can describe only a limited set of potential concept descriptions and does not include a description that perfectly classifies these databases' instances, which denote heart disease diagnoses collected from the Cleveland Clinic Foundation and the Hungarian Institute of Cardiology, respectively. Diagnoses are described by thirteen Boolean and numeric-valued attributes (e.g., sex, age, fasting blood sugar level). The target attribute is the Boolean-valued diagnosis indicating whether the corresponding patient has heart disease.

IB2 again significantly reduced storage requirements for both databases and also sacrificed classification accuracy. This occurred because the attributes used to describe the instances/diagnoses do not completely describe the target concept. This results with a large number of what appear to be exceptional instances. These instances resemble noisy instances since they are poor classifiers of similar instances. Therefore, IB2's accuracy suffered in these applications due to the same reasons it performed poorly in the noisy artificial domain applications.

Both algorithms performed terribly when applied to the Hungarian database. Their classification accuracies were below the frequency of the most frequent target value in this application (i.e., 63.9%). This occurred because this domain contains both imperfect attributes and a large number of missing attribute values. Figure 4.4 displays IB1's average learning curve (IB2's is similar) for this application. IB1's accuracy raised initially to about 80% and, surprisingly, trailed off slowly to about 56%. Section 4.3.3 will show that IB3 does not suffer this same fate. Section 5.2.1 explores whether alternatives to IB1's primitive algorithm for processing missing attribute values results with better learning behavior.

Noise-Free Databases:

The Congressional Voting database is linearly separable (Hampson, personal communication), which demonstrates that the attributes used to describe its instances (i.e., sixteen

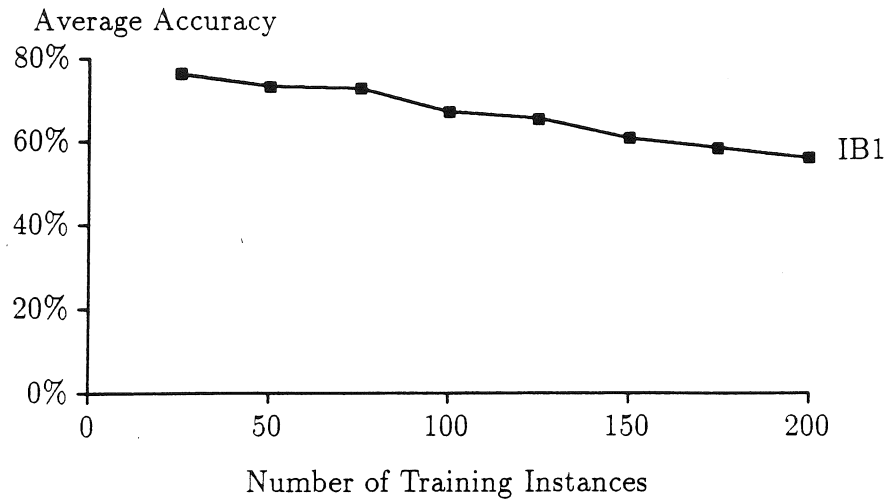


Figure 4.4: IB1's average learning curve for the Hungarian database.

Boolean voting records of the 1984 United States House of Representatives) are sufficient to correctly identify the target value (e.g., Democrat and Republican).

As expected, IB2 performed relatively better in comparison with IB1 than it did in the applications with noisy training instances. Although IB2 saved an average of less than 12% of the training instances, it still recorded an average classification accuracy that was closer to IB1's than in the previous applications. I observed similar behavior in experiments with several other databases with similar domain characteristics.

These results show that IB2 learns more slowly than IB1 even when the data is relatively free of noise. This occurred because IB2 is sensitive to the ordering of the training instances; it saves different instances depending on this order. If the ordering fools IB2 into saving a poor set of instances, then IB2's accuracy can suffer. IB3, which will be introduced in Section 4.3 to reduce IB2's sensitivity to noise, also reduces IB2's sensitivity to the poor instance orderings. For example, I re-ran both algorithms on this database but held the training and test sets constant. However, the order of the training instances was randomized for each of the 25 trials. IB2's accuracy varied from 83.8% to 93.8% with a standard deviation of 2.63 and a mean of 90.7%. IB3 accuracies varied from 90.0% to 96.2% with a standard deviation of 1.61 and a mean of 92.8%. IB2's range of accuracies (10.0%) was much higher than IB3's (6.2%), as was its standard deviation. (I found this behavior to be repeated in a similar experiment with the Cleveland database. IB2's range of accuracies (18.7%) and standard deviation (4.58) were again higher than IB3's (15.4% and 3.99).) Although this is only preliminary evidence, it appears that IB3 is more *stable* than IB2. That is, IB2 is more sensitive to malicious instance orderings than is IB3. Since IB1 should perform as well as or better than IB2 by saving all training instances for the relatively noise-free voting database, it is not surprising that IB2's average accuracy is lower than IB1's in this application.

Noisy Domains with Irrelevant Attributes:

The presence of irrelevant attributes slowed IB1 and IB2's learning rates in the LED and Waveform applications. The LED display domain was extended with 17 irrelevant Boolean attributes while the Waveform domain was extended with 19 irrelevant continuous-valued attributes with randomly assigned value having a mean of 0 and a variance of 1. The algorithms' accuracies decreased and IB2's storage requirements increased. These results were expected since the mathematical analyses predicted that instance-based learning algorithms require exponentially more instances to learn accurate concept descriptions with linear increases in instance space dimensionality.

However, IB1's performance suffered more than IB2's. This occurred largely because IB2's partial concept descriptions grew in size and consequently resemble IB1's more closely. Therefore, the difference between these algorithms' predictive performance decreased.

Summary:

In summary, IB2 significantly reduced storage requirements in each of the seven application domains. However, it also sacrificed classification accuracy, especially when the domains were noisy or had many exceptional instances (due to imperfect predictor attribute sets). Sections 4.3 and 4.4 describe additional experiments with these applications, including comparisons with the results generated by IB3, IB4, and an algorithm that builds decision trees.

Numeric Prediction Tasks

IB2 reduces storage requirements but sacrifices classification accuracy in symbolic prediction tasks. Its behavior is similar in numeric prediction tasks. Figure 4.5 provides some intuition to explain why this occurs. IB1 (with $k = 1$) would form the approximation denoted by the dashed line when given the ten instances shown as boxes in this one-dimensional instance space. The solid line represents the numeric target function. If the training set contains enough instances, then IB1 does not need to save many instances along subintervals with near-zero slope to ensure small errors for predictions of target values along them (e.g., the instance highlighted in the above figure can be removed). However, IB1 should retain all instances along subintervals where the absolute slope of the function is high. When $k > 1$, IB1 generates predictions by interpolating from stored instances. In such cases, it is not the target function's slope that determines which instances should be saved, but rather its acceleration (i.e., its second derivative); instances should be saved in areas of the instance space where the target function's values change quickly. Although IB1 is not told this information, it can be estimated by comparing (1) the similarity of a most similar saved instance i to a novel instance x with (2) the error resulting from using i to predict x's target value. If their similarity is high and the prediction error is low, then the accuracy of IB1's approximation

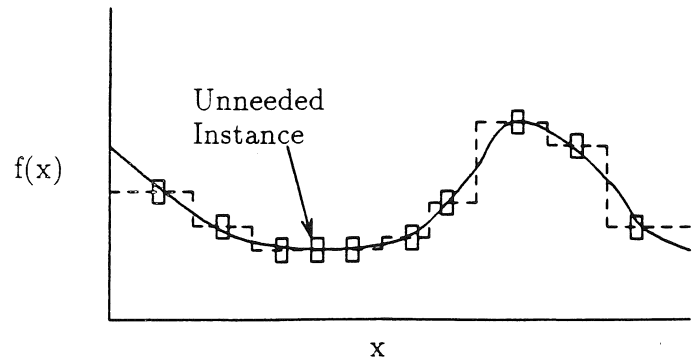


Figure 4.5: The highlighted instance is not a useful predictor.

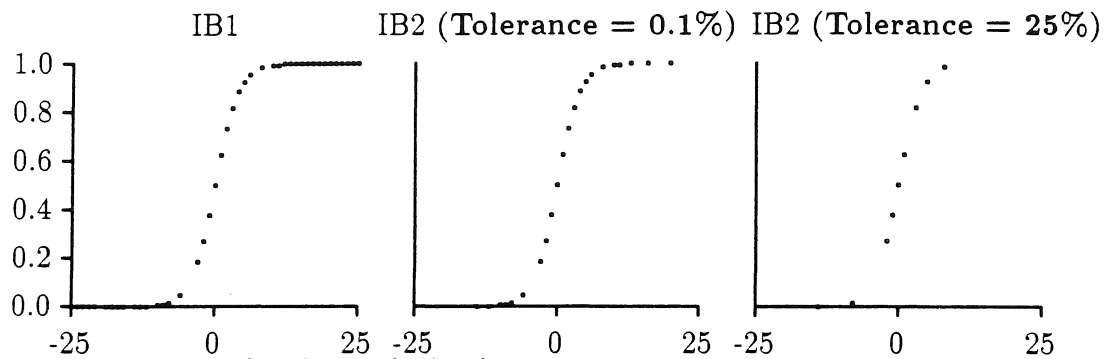


Figure 4.6: These graphs show which of 100 training instances were saved by these algorithms. The target function was $f(x) = 1/(1 + e^{-x/2})$ with no noise. IB2 saved only 22 and 9 instances when its error threshold was set to 0.1% and 25% respectively. On a disjoint test set, IB1's average relative error was 0.25%. The errors of the two versions of IB2 were 0.27% and 1.53% respectively.

would probably not be substantially increased by including this instance in the partial concept description. If we assume that this similarity is sufficiently high, then the prediction error alone can be used to determine whether the novel instance should be saved. This is exactly how IB2 decides which instances to save.

Figure 4.6 demonstrates IB2's capability to reduce storage without greatly sacrificing classification accuracy for numeric prediction tasks. This figure shows which instances from a 100-instance training set were saved by IB1 and IB2 (for two settings of IB2's error threshold) for a simple target function. IB2 safely discarded several instances along the two long subintervals where the target function's slope is zero. IB2 behaves conservatively, saving comparatively many instances, when the error threshold's value is low. Although higher settings decrease IB2's storage requirements, they also decrease its predictive accuracy. Informal

Table 4.4: Database characteristics.

Database Name	Training Set Size	Test Set Size	Number of Attributes	Mean	Standard Deviation
Hungarian Heart Disease	206	88	14	24.4	10.5
Cleveland Heart Disease	212	91	14	0.4	0.5
Horsepower	144	61	26	104.3	39.7
Number of Months	92	40	10	22.2	15.8
Cholesterol Level	212	91	14	246.7	51.8
Tumor Size	200	86	10	24.4	10.5

sensitivity experiments suggested that an tolerance_threshold setting of 15% is a good compromise value for IB2's only parameter. This setting is used in all the numeric-prediction applications of IB2 and IB3.

I applied IB1 and IB2 to six numeric prediction tasks, most of which involve medical diagnosis. The associated databases are briefly summarized in Table 4.4 and are detailed in more depth in Appendix A. As with the symbolic prediction experiments, 70% of the instances were used for training and the remaining 30% were used for testing. The purpose of these experiments is to examine the extent of IB2's tradeoff of predictive accuracy for reduced storage requirements in tasks involving functions from less contrived databases than the simple sigmoid function examined in the previous paragraph.

These six applications were chosen with care according to their unique domain characteristics (including distribution of values, mean and standard deviation). They are presented in order of increasing resemblance of their target function's distribution to a normally distributed curve. The first three functions have highly skewed distributions. The fourth has a spike in the beginning of its distribution, but otherwise resembles a normal curve. The target functions in the last two applications most closely resemble normal curves. These applications are described in more detail in the following list.

1. The first application involves the Hungarian heart disease database described earlier (Detrano *et al.*, 1989). The target function is degree of heart disease, but in this case there are only two values, 0 and 1, corresponding to no presence and presence respectively. Its distribution is polarized. This function was picked to see how the IBL algorithms perform when the instances contain the fewest possible distinct target values.
2. The second application concerns the same target function for the Cleveland heart disease database, which was also described earlier (Detrano *et al.*, 1989). However, in this case the target can have one of five possible values (i.e., the integers lie in the range $[0, 4]$). This function's distribution is also badly skewed; most of the diagnoses describe patients without signs of heart disease (value 0) and the fewest target values are for patients with extensive heart disease (value 4).

Table 4.5: Average relative error ± standard error and percent storage requirements.

Target function	IB1	IB2	Average Guess
Hungarian Heart Disease	0.39±0.022 100	0.43±0.026 36.7	0.46±0.002
Cleveland Heart Disease	0.19±0.005 100	0.20±0.005 52.9	0.25±0.003
Horsepower	0.04±0.002 100	0.06±0.003 24.9	0.14±0.006
Number of Months	0.20±0.006 100	0.22±0.006 57.4	0.24±0.004
Cholesterol Level	0.14±0.007 100	0.15±0.007 52.3	0.11±0.005
Tumor Size	0.22±0.004 100	0.23±0.004 62.0	0.17±0.003

3. The third target function, horsepower, is an attribute in the import automobile database (Schlimmer, 1987b). Its skewed distribution resembles one-half of a normal curve with a few outliers.
4. The fourth application is taken from Evlin Kinney's Echocardiogram database, which was donated by Miami's Reed Institute. The target function is the number of months that a patient survived after a heart attack. The predictor attributes include age, whether fluid has built up around the heart, and measures of contractability around the heart. Its distribution has a spike near 0 and is otherwise normally distributed with a large variance.
5. The fifth task is to predict the level of serum cholesterol for the diagnoses in the Cleveland database mentioned earlier. It has a relatively normal distribution, although it has several outliers.
6. The task in the final application is to predict tumor size. It is an attribute in the Breast Cancer database donated by the Ljubljana Institute of Oncology. Its values were discretized, which caused its distribution to more closely resemble a perfect normal distribution.

The experiments with the numeric prediction tasks had the same form as those for the symbolic prediction tasks. The instances chosen for the disjoint training and test sets were always randomly selected from the databases. The results, averaged over 25 trials, are summarized in Table 4.5. Average guess results, whereby the average of the training instances processed to date is used as the prediction, are listed alongside IB1 and IB2's results. In summary, IB2 again saved significantly fewer instances than IB1, but its predictive accuracy (in this case, relative error) increased slightly. The average guess method worked poorly for the target functions with skewed distributions but extremely well for those with relatively normal-shaped distributions. These results are examined in more detail in the following paragraphs.

Skewed Distributions:

IB2 saved less than 40% of the instances and sacrificed only 4% relative error for the Hungarian heart disease prediction task. Section 4.3.3 will show that, in fact, both IB1 and IB2 performed poorly here, mimicing their lackluster performance for the equivalent symbolic prediction task as described in Section 4.3.3 However, these IBL algorithms outperform average guess in applications with polarized distributions. Average guess works poorly in these situations since the average value lies far from both poles (i.e., 0 and 1 were the only target values in this application).

The IBL algorithms' performance improved when predicting the degree of heart disease for the Cleveland database diagnoses. As in all these applications, IB2's reduction in storage requirements cost a small amount in predictive accuracy (here about 1.5% relative error). The average guess algorithm's predictions were less accurate, mainly because there were a large number of values lying at one end of the target function's distribution while the others were dispersed in a half-normal curve. Thus, the average guess was distant from the majority of the target values in this function.

The results were similar for the horsepower target function, whose skewed distribution is also shaped like half of a normal curve. IB1's accuracy here was almost identical to the maximum accuracy recorded by CLASSIT (Gennari, 1990). IBL algorithms can learn more accurate approximations of this target function than it can for the others because it is relatively noise-free and its attributes are relevant to the target function. Not surprisingly, IB2's storage requirements were lower in this application, as they were for the sigmoid target function application with low levels of noise.

Normal Distributions:

The results for the task involving predicting the number of months that patients will live after suffering a heart attack are markedly different from those with the first three tasks. In this case, the average guess algorithm's error rate was lower than IB2's. It was also closer to IB1's error rate than in the previous applications. This occurred because the distribution of the target function approximates the normal, other than for a spike at its low end. For most instances, the average guess prediction should be accurate. In Section 5.2.2, I will show that IB1's accuracy improves dramatically when more than one instance is used to make predictions (i.e., $k > 1$).

Average guess performed clearly better than IB1 when predicting a patient's level of serum cholesterol. This occurred because the distribution of the target function is normal. However, I will show in Section 5.2.2 that IB1's accuracy is better than average guess's if IB1's setting for k is sufficiently large.

Average guess also outperformed IB1 when predicting breast cancer tumor size, whose values are also normally distributed. Figure 4.7 shows their learning curves for this target

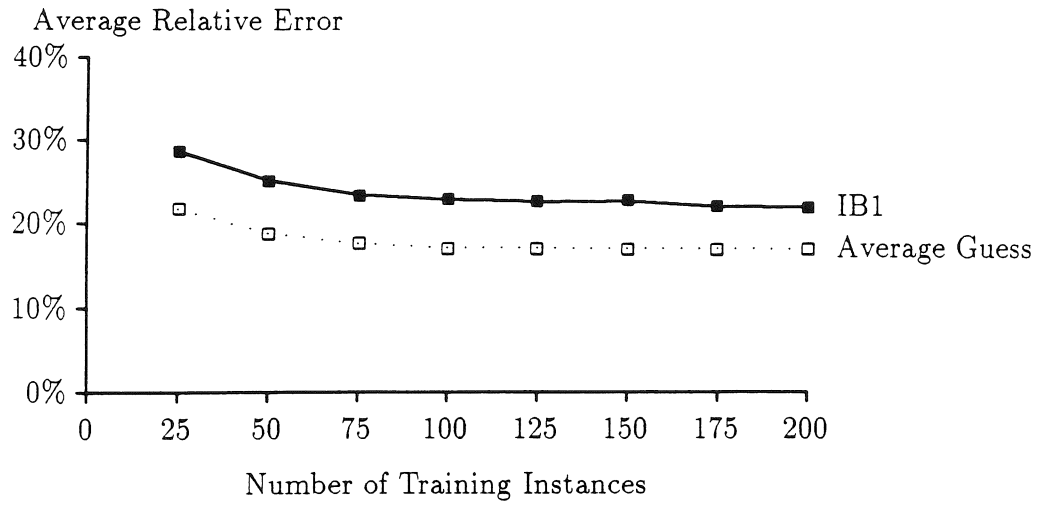


Figure 4.7: IB1's learning rate is slow for the tumor size application.

function. It is possible that, given enough instances, IB1's error rate might eventually reach average guess's error rate. However, its learning rate is obviously slower in applications to some database applications.

Summary:

As in symbolic prediction tasks, IB2 reduces IB1's storage requirements at the cost of increased predictive error. The experiments in this section also showed that IBL algorithms in general perform comparatively better than average guess when the target function's distribution is *not* normal. Although many functions are not normally distributed, this still bodes poorly for the numeric prediction capability of IBL algorithms. However, they can be improved in several ways. Section 5.2.2 shows that higher values for k yield more accurate prediction behavior.

Another way to improve the predictive accuracy of IBL algorithms is to restrict which instances are used to generate predictions. The next section demonstrates this approach and describes IB2's sensitivity to noise. Section 4.3 then shows how IB3 improves IB2's predictive accuracy by using a selective utilization filter (Markovitch & Scott, 1989) on the instances saved in the partial concept description.

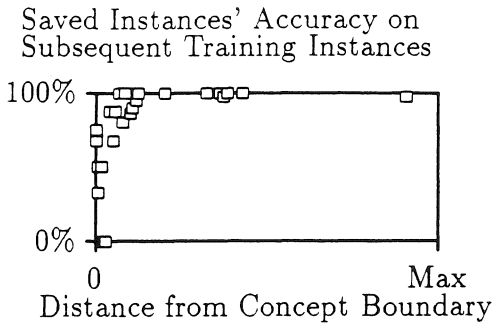


Figure 4.8: An instance's classification accuracy quickly reaches 100% with increasing distance from the concept boundary.

4.2.4 IB2's Behavior in the Presence of Noise

The Effect of Noise on Symbolic Prediction Tasks

An analysis of the predictive behavior of the saved instances shown in Figure 4.2 on page 64 shows that boundary instances can be distinguished from non-boundary instances according to their accuracy in nearest neighbor classification predictions for subsequently presented training instances. Figure 4.8 displays a scatter plot of these saved instance's classification accuracies versus their distance from the concept boundary. This plot shows that the accuracy of an instance rises quickly with its distance from the concept boundary. In fact, none of the instances saved by IB2 that were far from the concept boundary had low classification accuracies. However, this occurred only because the instances in the application domain were not corrupted by noise.

Figure 4.9 describes average results (over fifty trials) for IB1 and IB2 where the application domain is the same symbolic prediction task as shown in Figure 4.2 on page 4.2. IB2's reduction in storage requirements is most dramatic when none of the training instances are noisy. However, IB2 is more sensitive to the attribute noise level in the training set than is IB1, where the attribute noise level is defined as the probability that each attribute value (except the class attribute) of each instance is replaced with a randomly-selected value from the attribute's domain (according to the attribute's distribution). Given this definition for noise, the probability that a training instance will be misclassified by noise corruption is $(52.3 \times N)\%$ in this application, where N is the noise level. This is almost identical to the probability that noise in the class attribute will result with a misclassification, which is $(50.0 \times N)\%$. An investigation of the effects of either type of noise will show highly similar behavior. Therefore, is is sufficient to demonstrate IB2's behavior on only one type of noise

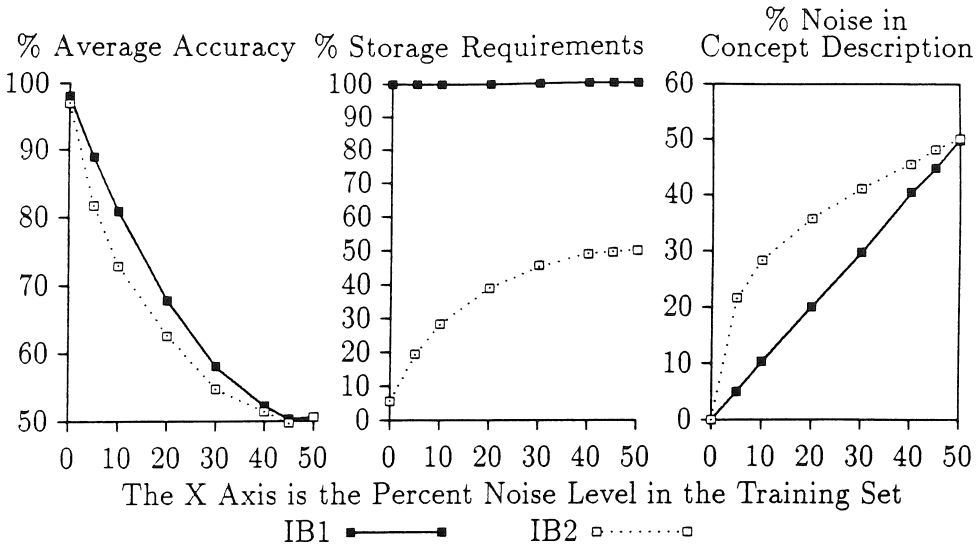


Figure 4.9: IB2 reduces storage requirements but is sensitive to noise.

(I chose to look at attribute noise). For other instance spaces, the probability of misclassifications in the training set can differ greatly, depending on whether all-attribute or class noise is used to corrupt the instances. In such cases, the type of noise with a higher probability of misclassification will result with accelerated degradations of IB2, but will not otherwise alter its relative behavior.

Figure 4.9 shows that IB2's classification accuracy decreases more quickly than does IB1's as the level of noise increases. This occurs because noisy instances are, naturally, almost always misclassified. Since IB2 saves only a small percentage of the non-noisy training instances, its saved noisy instances are more frequently used than IB1's to generate incorrect classification decisions.

Figure 4.9 also shows that, in this artificial domain, IB2's storage requirements increase quickly as the noise level increases and reaches an asymptote at 50%. This increase is greater than linear in the noise level because noisy instances will, with a high probability, be misclassified and, consequently, saved. These noisy instances will in turn misclassify non-noisy instances, which will also be saved. When approximately half the instances are noisy, then the accuracy of IB2's classification guesses approaches chance. About 50% of the training instances will be misclassified under these conditions, which explains the asymptote for IB2's storage requirements.

IB2's behavior in the presence of noisy instances can be clarified with a detailed example. Figure 4.10 summarizes IB2's behavior when the 250-instance training set was corrupted at the 10% noise level. In this case, twenty of IB2's 76 saved instances (26.3%) were noisy. This is not an unusual result; Figure 4.9 shows that an average of 28.3% of IB2's saved instances (at

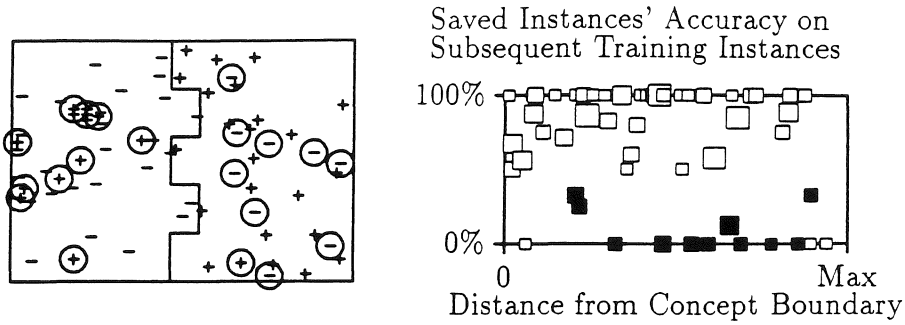


Figure 4.10: The figure on the left shows which training instances were saved by IB2 when the noise level was 10%. Noisy instances are circled and labeled as members ("+") or non-members ("–"). Those instances that participated in classification decisions are plotted at the right, where noisy instances are denoted by black boxes. Box sizes correspond to the number of classification decisions they made.

this noise level) were noisy. The scatter graph in Figure 4.10 shows that the noisy instances (black boxes) are easily distinguished from non-noisy instances (white boxes). The noisy instances invariably had poor classification accuracies on subsequently presented training instances, especially when they were located far from the concept boundary. However, a few (three, in this case) of the non-noisy instances also had poor classification accuracies. These classifiers were each used only once; they each misclassified a subsequently presented, noisy training instance. Their classification accuracies would have been higher if they were used to classify other instances. In summary, with sufficient numbers of classification attempts, non-noisy instances will have high classification accuracies. This will distinguish them from noisy instances located far from the concept boundary, which will have relatively poor classification accuracies.

The Effect of Noise on Numeric Prediction Tasks

IB2's numeric predictions are also sensitive to training set noise because the target values of a noisy instance and its classifying instance normally differ greatly. As before, most noisy instances are saved and they generate inaccurate predictions for subsequently presented non-noisy instances, which are then saved. Thus noisy instances increase IB2's storage requirements and decrease its predictive accuracy for numeric prediction tasks.

This behavior is exemplified in Figure 4.11, which displays which instances were saved by IB1 and IB2 when 250 training instances from the sigmoid application were subjected to attribute noise (i.e., there was a 20% chance that each instance's predictor attribute value was replaced with a randomly selected valued in $[-25, 25]$). Although IB2 saved only 103 of the 250 training instances in this application, 37 (36%) of its saved instances were noisy.

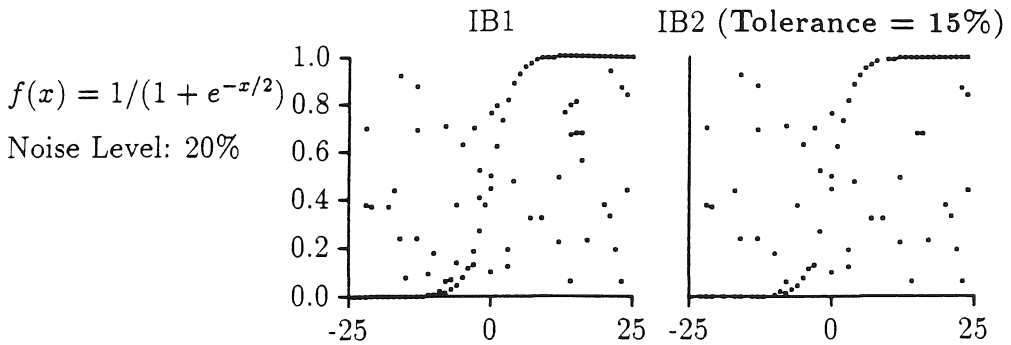


Figure 4.11: A larger percentage of the instances stored by IB2 were noisy (36%) than stored by IB1 (20%).

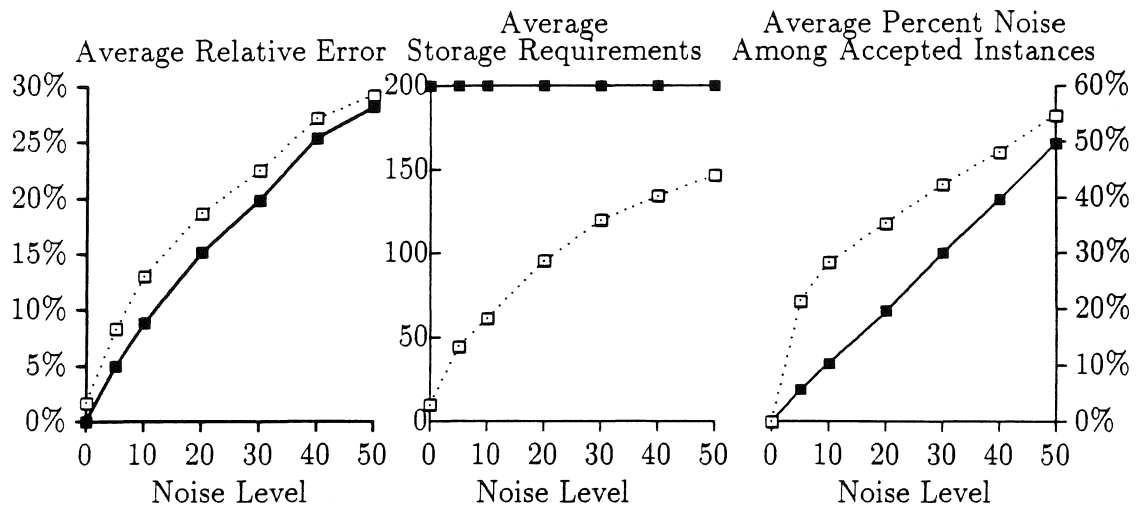


Figure 4.12: IB2 (dashed line) reduces IB1's (solid line) storage requirements but sacrifices some predictive accuracy. The target function is $f(x) = 1/(1 + e^{x/2})$ and these results are averages from 25 trials.

IB2's average relative error on a (noisy) test set was 14.8% whereas IB1's was only 12.0%. (IB2's accuracy with a low error threshold of 0.1% was also poor (14.3%).) As expected, the percentage of IB2's predictions for this application that were below the error threshold (i.e., 15% absolute error) was much lower for noisy instances (9.6%) than it was for non-noisy instances (80.4%).

Figure 4.12 summarizes the average results from applying the IBL algorithms to this sigmoid function, where each of the 25 disjoint training and test sets contained 200 and 100 instances respectively. The example shown in Figure 4.11 was typical of IB2's behavior; an average of 35.4% of the instances stored in IB2's partial concept description were noisy for the 20% noise level.

4.3 IB3: Tolerating Noise

The investigations described in Section 4.2 showed that, while IB2 significantly reduces storage requirements, it does not attain the same learning rate as IB1. Furthermore, experiments with several artificial domains and databases showed that IB2 is sensitive to noisy instances. This section introduces IB3, an extension of IB2 that increases its learning rate, decreases its sensitivity to noisy instances, and further reduces IB2's storage requirements.

4.3.1 Methods for Tolerating Noise

Several methods for tolerating noise have been published in the machine learning literature. Some of these succeed by testing whether perturbations of suspected noisy instances removes their noise. For example, MIRO's ECM (Error Correcting Module) locates clusters of instances that are suspected of being noisy and evaluates whether they are noisy be checking whether small perturbations in their attribute values would place them in a non-noisy cluster (Drastal, Meunier, & Raatz, 1989). Hirsh (1990) uses a similar technique for locating noisy instances. His algorithm generates a set of perturbations for each instance, constructs a version space for each perturbation, and then intersects the version spaces to reduce the effects of noise.

Most primitive empirical learning algorithms will overfit data containing noisy instances. Noise-tolerant extensions of these algorithms use statistical rather than analytic methods to reduce overfitting effects. For example, Clark and Niblett (1989) used a test of significance in an extension of the AQ15 algorithm (Michalski, *et al.*, 1986) that uses estimates of each rule's quality to determine whether it should be discarded. Similarly, Breiman *et al.* (1984), Quinlan (1986a), and Niblett and Bratko (1986) have all described non-incremental methods for post-pruning decision trees that discard nodes with low information content. These pruning methods all discard parts of a concept description that have little comparative information content and, because of this, may perform poorly during prediction attempts.

IB3's approach does not rely on the availability of domain-specific explanations, does not involve perturbing instances, and does not involve discarding abstractions. However, its method shares some similarities with methods developed to generate noise-tolerant rules and decision trees: it uses a test of statistical significance to discard parts of its concept description. However, IB3 is an *incremental* algorithm that uses a selective utilization filter (Markovitch & Scott, 1989) to determine which stored instances should be used to classify subsequently presented instances.

Table 4.6: The IB3 training algorithm.

Key: T: Training set
P: Set of predictor attributes
t: The target attribute
k: Number of most similar instances used
α & δ: Confidence thresholds for acceptance and dropping
PCD: Partial concept description

Train(T, P, t, k,tolerance_threshold,α,δ)
1. Set global variables
2. for each $x_i \in T$ do
   2.1 $x_i \leftarrow$ Pre_process(x_i, P)
   2.2 prediction $\leftarrow$ Performance(x_i, P, t, k,α)
   2.3 Learn(x_i,prediction,t,tolerance_threshold,δ)

Performance(x, P, t, k, α)
1. $S \leftarrow \emptyset$
2. $\forall y_i \in$ PCD: $S \leftarrow S \cup \{\langle y_i, \text{Similarity}(x, y_i, P)\rangle\}$
3. KSET $\leftarrow$ k_most_similar_acceptable_instances(S, k, α)
4. return Target_value_prediction(KSET,t, k)

Learn(x,prediction,t,tolerance_threshold,δ)
1. if ($|x_t -$ prediction$|$ > tolerance_threshold)
   then PCD $\leftarrow$ PCD $\cup \{x\}$
2. $S \leftarrow$ set of stored instances at least as similar as
   the k^{th} most similar acceptable instance
3. Update_prediction_records(S)
4. Discard_significantly_poor_instances(S, δ)

4.3.2 Description of the IB3 Algorithm

Table 4.6[4] details the IB3 training algorithm, which is a noise-tolerant extension of IB2. It has two additional inputs, denoted by parameters α and δ, that correspond to the level of confidence used to generate confidence intervals in IB3's test of significance. The parameter α represents the level of confidence used to generate intervals when IB3 evaluates whether an instance is *acceptable* and the parameter δ is used when IB3 decides whether to discard an instance from a partial concept description. IB3's testing algorithm is identical to IB2's except that the k most similar *acceptable* stored instances are used to derive target value predictions for test instances.

[4]The sub-functions not described in this section were previously described in Table 2.1 on page 21.

IB3's training algorithm is identical to IB2's except that it employs a "wait-and-see" evidence-gathering attitude to determine which of the saved instances are expected to accurately predict similar instance's target values. Its pre-processing and similarity functions are identical to IB2's. However, its prediction-generating and memory-updating components differ as follows:

1. *Prediction function*:

   (a) Instead of using the k most similar instances for generating a prediction, IB3 uses the k most similar *acceptable* instances for this purpose.

   (b) If only $k' < k$ stored instances are acceptable, then IB3 generates a prediction from these k' instances and the most similar non-acceptable $k-k'$ instances. If less than $k - k'$ non-acceptable instances have been saved, then all the non-acceptable instances are used to help generate predictions.

2. *Memory updating function*:

   (a) IB3 maintains a *prediction record* with each saved instance y. This is a record of the number of correct and incorrect prediction attempts made by y (i.e., when simulating $k = 1$), where an attempt's success is determined by a comparison of y's prediction error and the Tolerance_threshold (as is done to determine whether to store a training instance). A prediction record summarizes an instance's classification performance on subsequently presented training instances. IB3 uses this record to predict how well it will perform in future prediction attempts.

   (b) After each prediction attempt, Update_prediction_records will update the prediction records for the set of instances that are at least as similar as the most similar acceptable instance to the instance being classified. If none of the instances are acceptable, then IB3 simulates what would have happened had there been at least k acceptable instances. First, it generates a random number r between 1 and the number of stored instances. It then updates the prediction records of the most similar r stored instances. This process is depicted in Figure 4.13. If none of the saved instances are acceptable, then y in this figure denotes the r^{th} most similar instance to x.

   (c) IB3 employs a significance test to determine which of its saved instances are acceptable classifiers, which ones are believed to be noisy, and which currently appear to be neither. Only acceptable instances are used to classify subsequently presented instances. Noisy instances are discarded from the concept description by Discard_significantly_poor_instances. All other stored instances are retained but not used to generate predictions.

In summary, IB3's policy for updating prediction records is such that it updates the records of all instances that lie within a hyper-sphere in the instance space whose radius is the distance between the instance being classified and the k^{th} most similar instance. If between

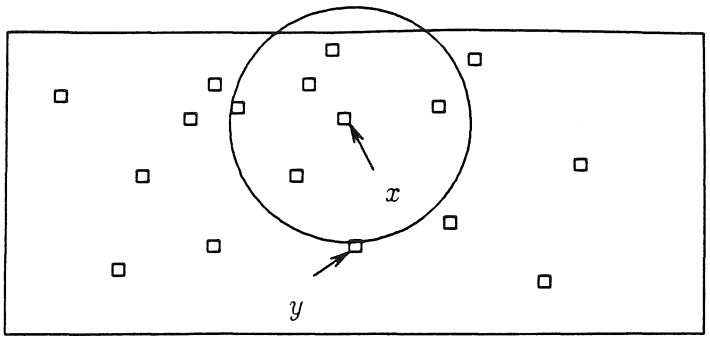


Figure 4.13: If the instance to be classified is x and its k^{th} most similar accepted instance is y, then the circle shown delineates the set of instances whose prediction records will be updated.

1 and k instances are acceptable, then this radius is the distance to the least similar acceptable instance. A similar hyper-sphere is generated when none of the saved instances are acceptable.

An acceptable instance is one that, after being stored, displays significantly accurate predictive behavior. An instance y's predictive behavior is defined as its number of successful versus total prediction attempts. That is, y's prediction record on those subsequently presented instances x that it was at least as similar to as was the kth most similar stored instance. This value can be interpreted as an estimate of the conditional probability

$$\Pr_x(|\text{ prediction } - x_t| \geq \text{tolerance\_threshold} \mid \text{Similarity}(x, y, P) \text{ is high})$$

This conditional probability is compared with the unconditional probability

$$\Pr_x(|\text{ prediction } - x_t| \geq \text{tolerance\_threshold})$$

to determine whether an instance is acceptable, noisy, or neither. IB3 accepts an instance y if y's conditional probability is significantly greater than the unconditional probability. (Notice, however, that the unconditional probability is a parameter of y.) The instance is deemed noisy and is removed from the concept description if its predictive accuracy is significantly less. A *confidence intervals of proportions test* is used to determine whether an instance is acceptable, mediocre, or noisy. Confidence intervals are constructed around both the estimated conditional and unconditional probabilities using

$$\frac{p + z^2/2n \pm z\sqrt{p(1-p)/n + z^2/4n^2}}{1 + z^2/n}, \tag{4.1}$$

where p out of n samples have succeeded (for the test being evaluated) and z, which can be read from a normal-curve area table, refers to the confidence level desired (Spiegel, 1988).

The following conservative test for statistical significance is used.[5] If the lower endpoint for the interval generated for the conditional probability is greater than the higher endpoint generated for the unconditional probability, then the instance is acceptable. Similarly, instances are dropped when their conditional probability interval's higher endpoint is less than their unconditional probability interval's lower endpoint. Finally, if the two intervals overlap, then a decision on whether to accept or discard the instance is postponed until after further training. Again, only *acceptable* instances are allowed to participate in prediction decisions.

The interpretation of the conditional and unconditional probabilities differ depending on whether the prediction task involves symbolic or numeric target values. When predicting symbolic values, the conditional probability is y's number of correct classifications divided by its number of classification attempts. Its unconditional probability is the observed relative frequency of its class. For example, if the task is a binary classification task and y is a negative instance, then its unconditional probability is the number of negative instances processed to date divided by the number of total instances processed to date. When constructing the interval for the instance's classification accuracy, p is the instance's observed accuracy and n is the instance's number of classification attempts. When constructing the interval for the instance's class's frequency, p is the class's observed frequency and n is the number of processed training instances. The value for z is obtained from a standard table of normal distributions corresponding to the selected confidence level to be used to generate the confidence intervals.

The interpretations of these estimated probabilities are more complex when the task is to predict numeric values. Although the same definition for is used for instance acceptability, the unconditional probabilities must be estimated differently since class frequencies do not exist. IB3 instead uses a method similar to coarse-coding (Rumelhart *et al.*, 1987); the target function's dimension is partitioned into a set of equal-sized and overlapping *target intervals.* The estimated unconditional probability, for some saved instance y, is the percentage of processed training instances whose target values are in the target interval that most closely centers y's target value. That is, the unconditional probability is

$$\Pr_y(x_t \in \text{Centering\_interval}(y_t)) \tag{4.2}$$

where x_t and y_t are those instance's target values. Estimated conditional probabilities are defined as the percentage of similar instances that have target values in this same target interval, which is expressed as

$$\Pr(x_t \in \text{Centering\_interval}(y_t) \mid \text{Similarity}(x, y, P) \text{ is high}). \tag{4.3}$$

[5] This significance test assumes that the number of successes (i.e., classification accuracies or class frequencies) are binomially distributed. While this assumption may not be correct, several of my IBL algorithms that employ this significance test have performed well in many applications. Some of these applications will be described later in this section.

Accepted instances are those whose accuracy's confidence interval is above the confidence interval constructed around the frequency with which instances are indexed into its centering target interval.

The advantage gained by comparing a saved instance's accuracy with its class's observed relative frequency (or its target interval's frequency) requires further explanation. IB3 normalizes the acceptance of an instance with respect to these frequency distributions to decrease its sensitivity to skewed distributions. Instances in concepts/intervals with high observed relative frequencies are expected to have relatively high classification accuracies since a relatively high percentage of its possible prediction attempts will be for instances that are similar to it. Similarly, instances in concepts/intervals with low observed relative frequencies are expected to have relatively low classification accuracies. By comparing an instance's accuracy against its class's frequency, IB3 can more easily tolerate skewed concept/interval distributions.

The preceding discussion on confidence intervals lends little intuition concerning how they are altered during training (i.e., over time). Confidence intervals are initially wide. They shrink monotonically in width with the number of classification attempts. Figure 4.14 details how the confidence intervals for an instance's accuracy and its class's frequency changes

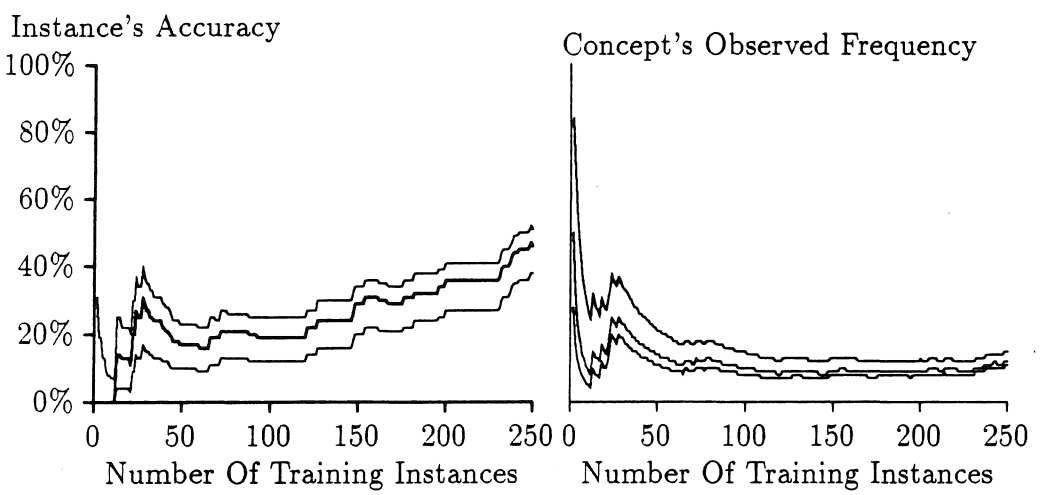


Figure 4.14: Confidence intervals shrink in width over time. This is an example of an accepted (good) instance in the LED-7 Display domain. The middle curve in each graph denotes the actual value. The other two curves in each figure denote the respective confidence interval's upper and lower bounds.

during training. This instance, taken from the LED-7 Display domain, was accepted since its classification accuracy, which reached 46%, is significantly better than its class's observed relative frequency, which reached 12%. This training instance was not accepted until after 120 training instances were processed. At that point, the accuracy interval's lower bound

first became higher than the frequency interval's upper bound. The instance's accuracy remained significantly higher than its class's frequency for the duration of training.

Both confidence intervals are initially large in width and gradually shrink. The frequency interval is smaller in size because it is computed over the entire set of observed training instances (250 total), whereas the accuracy interval is instead computed only over this instance's classification attempts (i.e., 53). The accuracy interval's upper endpoint and the concept frequency interval's lower endpoint are closer to the actual values than are the other two endpoints. This is because IB3 used a looser bound (75% confidence level) to determine which instances to drop than the bound it used to determine which instances to accept (90% confidence interval). These parameter settings were chosen to make it difficult for instances to be accepted. Thus, a high confidence level was selected for the acceptance parameter. However, a lower confidence level was selected for dropping to encourage IB3 to drop those instances with even moderately poor classification accuracies. These are the only two additional parameters required by IB3 for symbolic prediction tasks and these settings are used in all the symbolic prediction experiments with IB3 (and IB4) unless otherwise noted. An analysis of IB3's sensitivity to the settings of these parameters is described at the end of Section 4.3.3.

However, a lower setting (75%) was used for acceptance in the experiments with numeric prediction tasks. Informal sensitivity experiments indicated that this setting gave the best results for IB3 in these experiments. The values for IB3's other parameters are also fixed across all the numeric-prediction experiments described in this chapter. IB3's error threshold parameter value is set to 15% (as is IB2's), which also determines the (normalized) length of the target intervals (0.3). Finally, 20 target value intervals are used to partition the target function's range.

4.3.3 An Examination of IB3's Learning Behavior

This section examines IB3's learning behavior in applications to the same artificial domains and database applications used to evaluate IB1 and IB2 in Section 4.2.3. The following sections investigate IB3's behavior on symbolic and numeric prediction tasks respectively.

Symbolic Prediction Tasks

IB3 assumes that the prediction records of noisy instances will distinguish them from non-noisy instances. Noisy instances will have poor prediction accuracies because their nearby neighbors in the instance space will invariably have other classifications. This assumption is exemplified in Figure 4.15, which summarizes IB3's performance on the same data set given to IB2 to produce Figure 4.10 on page 78. Their are several striking differences between

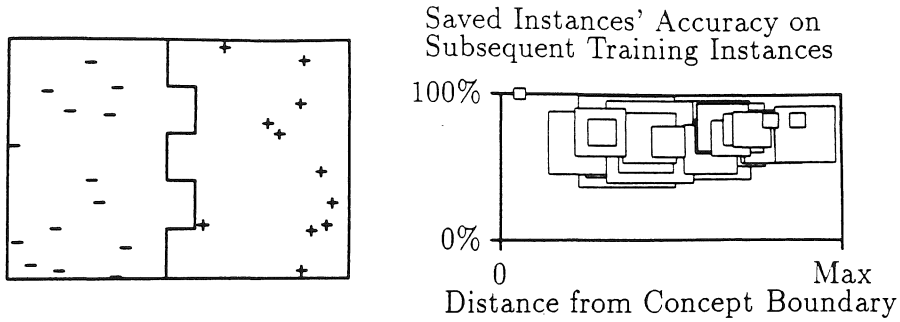


Figure 4.15: IB3's saved instances and their scatter plot of accuracy versus distance from concept boundary when applied to the same training set as was IB2 to produce Figure 4.10 on page 78. No noisy instances were used by IB3 for classification purposes (IB2 used 20). Box sizes correspond to the number of classification attempts they made.

these two figures. First, IB3 used *zero* noisy instances for prediction tasks whereas IB2 used twenty. This provides the first piece of evidence that IB3's selective utilization filter works in noisy domains. Second, IB3 used only 25 instances to generate predictions whereas IB2 used 76. Third, IB3's acceptable instances recorded relatively high classification accuracies. This is another consequence of IB3's noise-tolerant filter. Next, several of IB3's acceptable instances were frequently used for prediction (as denoted by the large sizes of the boxes in the scatter plot in Figure 4.15) because fewer instances were used to classify subsequent training instances. Finally, IB3's acceptable instances, in comparison to IB2's set of saved instances, are located relatively far away from the concept boundary. This occurs because IB3 accepts only those saved instances with significantly good classification accuracies and, in the limit, non-boundary instances will always have higher classification accuracies than near-boundary instances.

In summary, Figure 4.15 shows that IB3 displays superior learning performance on this training set both in terms of storage requirements and in its ability to filter noise from the partial concept description. In fact, this example is typical of IB3's performance on this artificial domain. Figure 4.16 summarizes IB3's learning performance in comparison with IB1 and IB2. These results are averaged from 50 learning trials. The independent measure was the level of attribute noise in the artificial domain. The dependent measures were classification accuracy, storage requirements, and the percentage of noisy instances in the partial concept description. When the training set was corrupted at the 10% noise level, IB3 used an average of only 5.8% of the training instances for its classification judgments. In contrast, IB1 and IB2 used 100% and 28.2% of their training instances, respectively.

Figure 4.16 also shows that IB3's storage requirements approached zero at high noise levels. This occurred because few instances have good prediction accuracies when the noise level is high. In the extreme case, none of the saved instances will have good prediction accuracies. When this occurs, IB3 will not accept any of the saved instances.

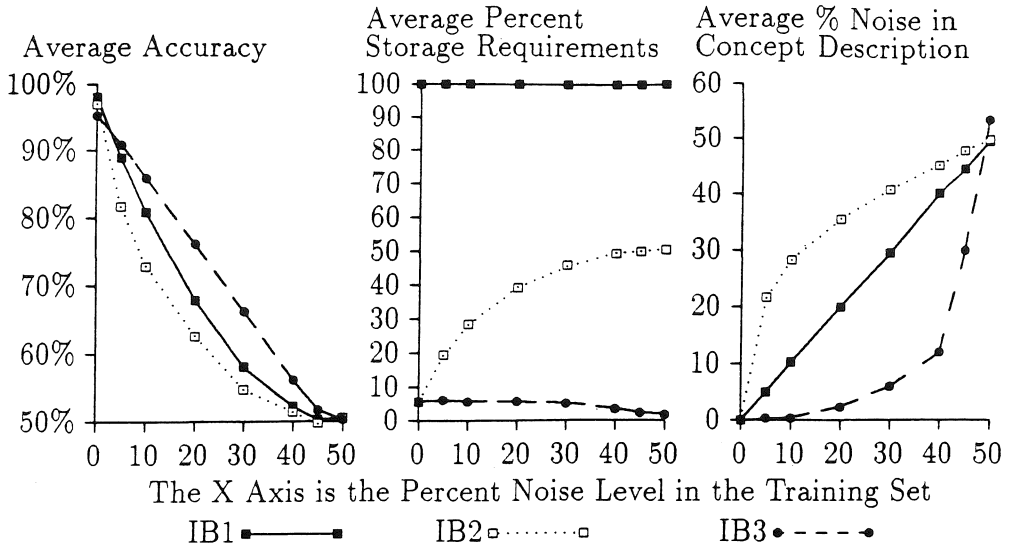


Figure 4.16: IB3 filters noise better than either IB2 or IB1.

IB3 successfully detected and eliminated noisy instances from the partial concept description. Its percentage of noisy instances in the partial concept description remained low until the 45% noise level. Since IB3 used few noisy instances to classify subsequently presented training instances, its classification accuracy degraded more slowly with the noise level than did the accuracy of the other algorithms.

In summary, IB3 performed well on a simple artificial domain, successfully preventing most of the noisy instances from participating in prediction decisions. IB3 recorded higher classification accuracies and lower storage requirements than did IB1 and IB2 when the training instances were corrupted with noise. However, its classification accuracy was slightly lower when no noise existed in the training set. This occurred because IB3 mistakenly suspected that some near-boundary instances with poor classification accuracies were noisy. This problem is intensified when the application domain contains several exceptional instances that form small disjuncts, each located far from the others in the instance space. Therefore, IB3's benefits are most noticeable in applications with noisy domains in which the vast majority of instances have the same classification as their neighbors. Otherwise, IB3's learning rate will be slightly slower than IB2's.

Although IB3 performed comparatively well on this artificial domain, this does not ensure that IB3 will continue to do so in other applications. Therefore, it was applied to the same set of databases (including training and test sets) used to evaluate IB1 and IB2 in Section 4.2.3. Table 4.7 summarizes the results. As expected, IB3 outperformed IB2 in all seven applications. The results for C4, Quinlan's (1987) extension of ID3 that post-prunes

Table 4.7: Percent Accuracy ± Standard Error and Percent Storage Requirements. IB3 recorded higher classification accuracies and lower storage requirements than IB2. In most cases its classification accuracy also compares favorably with IB1's and C4's.

Database	IB1	IB2	IB3	C4
LED-7 display	71.6±0.4 100	63.0±0.9 41.6	72.5±0.4 20.1	68.9±0.5
Waveform-21	75.5±1.1 100	68.4±1.1 32.3	74.2±1.2 11.1	71.5±1.2
Cleveland	75.1±0.8 100	71.4±0.9 32.0	78.4±0.9 3.9	75.2±1.2
Hungarian	56.1±2.2 100	53.1±2.4 36.9	79.4±0.9 4.3	77.6±0.9
Voting	91.8±0.4 100	90.9±0.5 11.6	90.6±0.6 3.5	96.1±0.6
LED-24 display	47.9±0.6 100	43.7±0.8 60.1	46.6±0.7 25.3	66.9±2.1
Waveform-40	68.6±0.7 100	64.0±0.7 38.3	67.2±1.1 11.8	70.9±1.0

decision trees, are also presented.[6] C4 achieved high accuracies on real-world medical applications (Quinlan, Compton, Horn, & Lazurus, 1986), recorded dramatically faster learning rates than genetic algorithms on multiplexor learning tasks (Quinlan, 1988), and has generally performed well on a large and varied set of learning tasks that are similar to the ones chosen for these experiments (Quinlan, 1987). Furthermore, C4 is highly similar to a set of decision tree algorithms that have performed well in a large number of research experiments and industrial applications (Breiman *et al.*, 1984; Michie *et al.*, 1984; Cestnik *et al.*, 1987) Therefore, C4's results provide a good comparison for judging the accuracy of the IBL algorithms' predictions. The remainder of this section describes IB3's results in more detail.

Noisy Artificial Domains:

IB3's significantly outperformed IB2 in the experiments with the noisy LED-7 Display and Waveform-21 domains, where it recorded higher classification accuracies and substantially lower storage requirements. This was expected since the same qualitative results occurred in the experiments with the simple noisy artificial domains described earlier. These experiments also correctly anticipated that IB3's accuracy would be higher than IB1's on the noisy LED-7 domain. Since all the attributes in this domain have the same (Boolean) range, IB1 becomes identical to the nearest neighbor (NN) algorithm in this application. Breiman (*et al.*, 1984) reported that the NN algorithm recorded a 71% classification accuracy on 5000 test instances (in a single learning trial). IB1's accuracy (71.6%) is, as expected, highly similar. C4's accuracy (68.9%) was similar to the C4 decision tree learning algorithm's accuracy (70%) on a single learning trial (Breiman *et al.*, 1984). IB3's slightly faster learning rate (72.5% after 200 instances, where 74% is the Bayes optimal accuracy) is due to its learning bias; the LED-7 domain can more easily be described by a set of prototypical instances than by decision trees. This is because the instances in each concept are normally distributed

[6]Storage requirements are not presented for C4, which was tested only to judge whether the IBL algorithms' accuracies were similar to those obtainable by more popular machine learning algorithms.

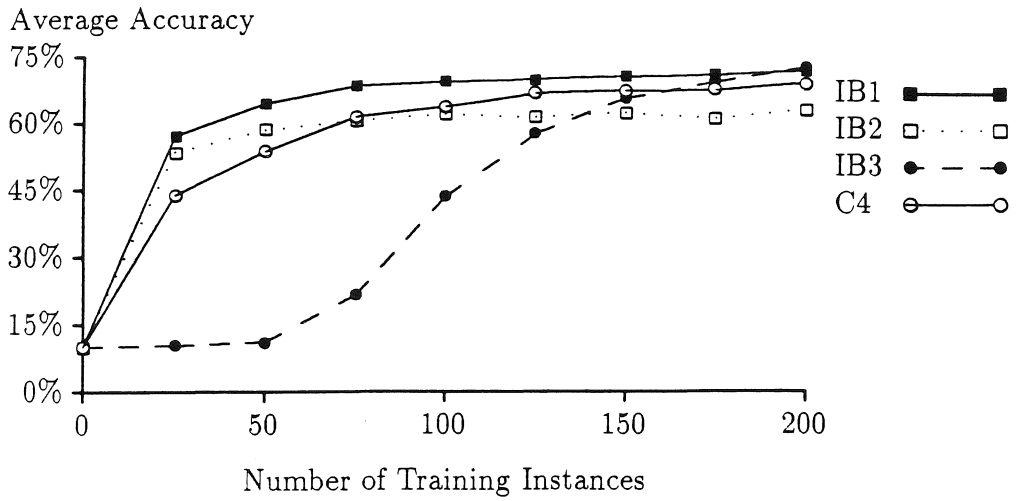


Figure 4.17: Learning curves for the LED-7 application.

around a single prototype. Most correct classification judgments will follow once each prototypical instance is saved and used for classification predictions. Figure 4.17 shows that IB1 does indeed learn the LED-7 concepts more quickly than does C4. IB3 learns far more slowly since it requires statistical evidence that an instance is a significantly good classifier before it is used in prediction decisions. This is a much more time-consuming process than C4's requirement that interior nodes contain significantly informative classification information.

The Waveform-21 domain adds an additional problem to the learning task; not all attributes are equally relevant for the purposes of prediction. As evidence, only eight of the 21 predictor attributes were required in a CART-generated decision tree to achieve an accuracy of 72% (Breiman *et al.*, 1984), which I found to be approximately the same accuracy recorded by C4 (71.5%). Furthermore, IB3's learning rate will be slower than IB1's in the presence of irrelevant attributes because a higher percentage of instances with different classifications will be among those that seem similar to each saved instance, thus making it more difficult for them to achieve acceptability. Therefore, IB3 does not outperform IB1 in terms of classification accuracy in this domain, even though it is noisy. However, both algorithms outperform C4 and CART in this application. This is again due to the IBL algorithms' less restrictive learning bias; the concept descriptions for this application are more easily expressed using IBL's piecewise-linear approximations, which are not restricted to being orthogonal to the given attribute dimensions. Subsequently, the learning rates of algorithms that build decision trees is sufficiently decreased so that they record lower classification accuracies than IB3 in this application.

Interestingly, Breiman and his colleagues reported that the NN algorithm recorded a surprisingly high accuracy on this domain (78%). However, this result was for a single learning trial and might be due to the choice of training instances in the training set. A more

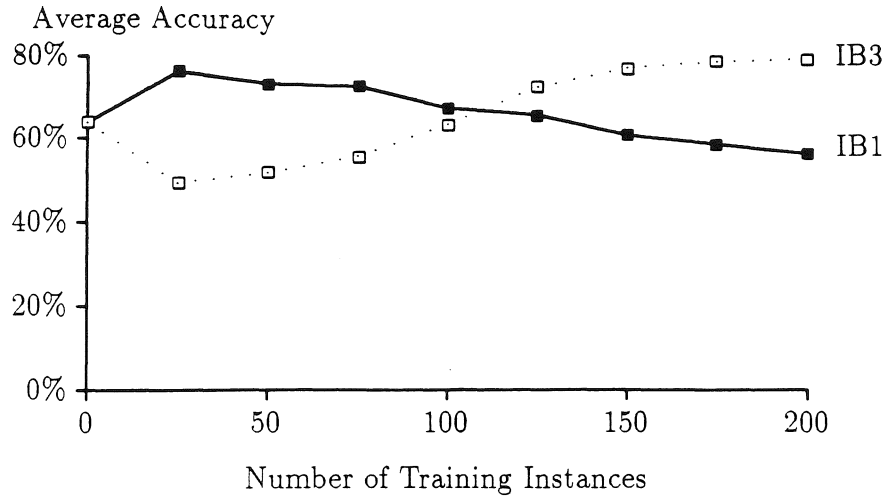


Figure 4.18: IB1 and IB3's average learning curves for the Hungarian database.

plausible reason for NN's outstanding accuracy here is that, by good fortune, the attribute values deemed most relevant by NN for the waveform-21 domain happen to closely correspond to those attributes that *are* the most relevant for predicting classifications. Since IB1 does not normalize the values of numeric-valued attributes, its similarity function (Euclidean distance) assumes that the attribute with the highest range of numeric values has the most relevance (assuming other factors remain constant). The most relevant attributes for the Waveform-21 domain are in fact those with the largest range of values. Unfortunately, the three IBL algorithms did not receive this information and, therefore, have slower learning rates than NN for this application. Section 4.4 reports that IB4 is able to uncover most of this information, which allows it to record faster learning rates than IB1, IB2, and IB3 in this application.

Databases with Imperfect Attribute Sets:

IB3's advantages are far more pronounced in the applications to the two heart disease databases, where it recorded the highest average classification accuracy and the lowest average storage requirements among the IBL algorithms. In fact, less than 5% of the instances for these databases are required to achieve these high accuracies. IB3's results are most striking for the noisy Hungarian database, where its average accuracy is 20% higher than IB1's and IB2's. Figure 4.18 shows that IB3's average learning curve, which first dips to random choice when none of the instances are acceptable, eventually rises far above IB1's. After 200 instances, IB3's accuracy is significantly higher than IB1's ($t(24) = 2.02, p < 0.05$). This is an excellent example of a situation in which instances should prove their utility before being allowed to partake in classification decisions. C4's test of statistical significance also helped it achieve a relatively high accuracy. However, IB3's more relaxed learning bias again allowed it to perform favorably in comparison.

Noise-Free Database:

The results with the experiments with the noise-free artificial domain suggest that IB3's learning rate will be slower than IB1's in some noise-free situations. This proved true in the experiment with the relatively noise-free Congressional Voting database. IB3's performance improvement over IB2 is not substantial in this application. This was predicted since they performed equally well under similar conditions, as described in Section 4.2.3 (i.e., when applied to the 2-dimensional artificial domain at the 0% noise level). C4 outperformed the IBL algorithms in this application because it found that few attributes were required (i.e., between 1 and 5 attributes) to yield accurate trees. This was also expected since this domain is linearly separable (Hampson, personal communication).

Although IB3 sacrificed classification accuracy in this application, its storage requirements were still extremely low (i.e., IB3 accepted an average of only 3.5% of the training instances).

Domains with Irrelevant Attributes:

IB3's classification accuracy on the LED-24 domain is comparatively terrible: it was 27.4% *below* the Bayes optimal limit and about 20% below C4's accuracy. C4's results are useful because they show that IB1 and IB2 *also* performed poorly in this application. This occurred because these IBL algorithms do not perform well in applications with multiple irrelevant attributes. Breiman (*et al.*, 1984) reported that the NN algorithm achieved a 41% accuracy on this domain, which appears plausible considering that IB1's accuracy was only a 6.9% higher. However, the reliability of this figure must be questioned since the algorithm was tested on only one training set, which may not have been highly representative.

CART achieved a 70% classification accuracy on this domain (Breiman *et al.*, 1984), which is somewhat similar to C4's here (66.9%) (again, this discrepancy is probably due to the single learning trial problem). The decision tree algorithms outperform the IBL algorithms in this application because they measure the information content of each attribute, which allows them to discover which attributes are irrelevant. I will later show that IB4 (Section 4.4) can also discover this information.

The situation with the Waveform-40 domain is not much better; C4 outperforms all three IBL algorithms (CART's accuracy was 72%). Surprisingly, Breiman reported that the NN algorithm's classification accuracy was only 38% on this domain, which is far lower than IB1's classification accuracy (68.6%). Either the accuracy of the NN algorithm was incorrectly reported (i.e., the classification accuracy was reported rather than the misclassification rate) or the single learning trial was not sufficient to accurately determine NN's capabilities on this domain. Therefore, I decided to test NN on this application. It's average classification accuracy on these same set of 25 training and test sets was 66.4%. The accuracies of the 25 tests varied from 55% to 74% with a standard deviation of 5.3. Although it is possible, the probability is low that a randomly chosen training and test set would yield a 38% accuracy.

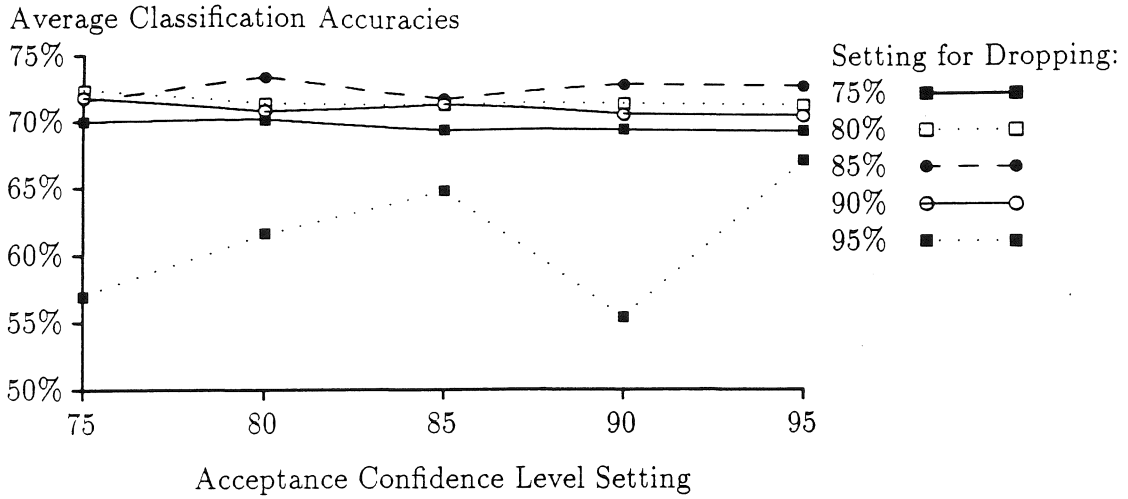


Figure 4.19: Many settings for IB3's parameters resulted in good performance.

Therefore, I believe that their reported figure was actually NN's correct classification rate rather than its misclassification rate. Nonetheless, these results show that IBL algorithms will not perform well in domains with numerous irrelevant attributes without suitable extensions that allow them to determine the relative relevances of attributes.

Summary:

In summary, IB3 demonstrated significant improvement over IB2 in these applications. It consistently recorded lower storage requirements and higher learning rates. IB3 also compared favorably with IB1 in measurements of classification accuracy. Finally, IB3's relatively weak learning bias for piecewise-linear approximations of concept descriptions often allowed it to achieve slightly faster learning rates than C4 in these applications. However, the results with domains containing high numbers of irrelevant attributes exposed the fact that IB1, IB2, and IB3 do not perform well when instances are described by many irrelevant attributes. This observation was anticipated by the mathematical analyses in Chapter 3, which predicted that IB1 requires exponentially more instances to achieve similar accuracies with linear increases in dimensionality.

It is possible that IB3's performance in these applications is highly sensitive to its parameter settings (i.e., the confidence levels for accepting and dropping instances). This hypothesis was tested in a sensitivity analysis using a factorial design, where the two parameters were varied from 75% to 95% in 5% increments. The test application was the LED-7 domain and the results were averaged over five trials for each pair of settings. Figure 4.19 shows that the settings chosen for IB3, 90% for acceptance and 75% for dropping, do not yield ideal learning behavior. Furthermore, it appears that IB3's behavior in this application is relatively robust with respect to these two parameters' settings. However, settings below

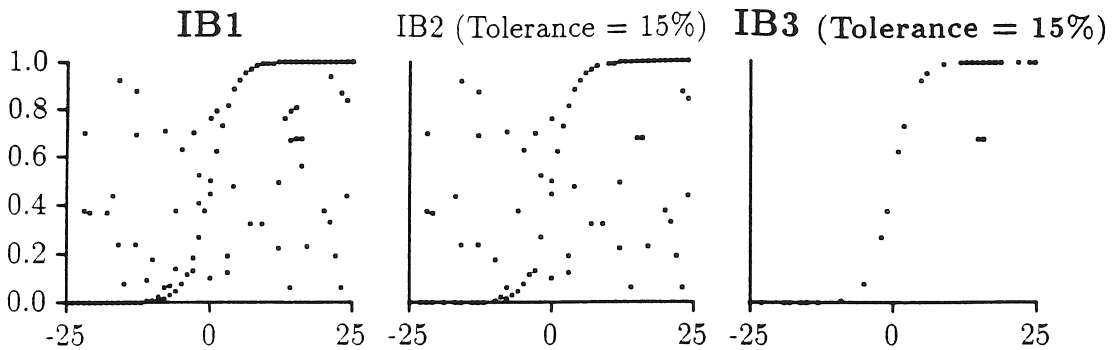


Figure 4.20: These graphs show which instances were saved by the three algorithms in an application to the $f(x) = 1/(1 + e^{-x/2})$ target function at the 20% attribute noise level. IB3 accepted only two noisy instances in this example.

80% or above 90% for acceptance begin to deteriorate IB3's performance. Although not shown in the figure, this experiment also indicated that IB3 will accept more instances as the confidence level for dropping increases. At the 90% acceptance level, the 75% dropping level resulted with the lowest storage requirements (average storage requirements ranged from 56.0 to 70.2 instances as the confidence level for dropping instances was raised from 75% to 95%). Although these were the parameters settings chosen for testing IB3, several other settings resulted with similar average classification accuracies and storage requirements for this application. However, this is not true for all applications, as demonstrated in another sensitivity analysis in Section 4.5.

Numeric Prediction: An example

IB3 also improves IB2's performance on numeric prediction tasks. This section examines IB3's learning behavior in an application to a simple numeric prediction task that was used in Section 4.2.3 to evaluate IB2's learning behavior.

Figure 4.20 summarizes the first piece of evidence that IB3's benefits are also noticeable in numeric prediction tasks. This figure is an extension of Figure 4.11 on page 79. It shows which instances were accepted by IB3 alongside those saved by IB1 and IB2 when the attribute noise level was 20% and the application is the simple sigmoid function. Only two of the 37 instances accepted by IB3 were noisy. That is, only 5.4% of IB3's accepted instances were noisy whereas the percentage of noisy instances saved by IB1 and IB2 were 20.0% and 35.9% respectively on this example. IB3's mean error on this training set was 8.2%, far better than IB1's 12.0% and IB2's 14.8%. Figure 4.21 is an extension of Figure 4.12 on page 79 that also includes IB3's averaged results (from 25 trials). (Note how closely this resembles the averaged results with the simple symbolic application, which is shown in Figure 4.16 on page 88. IB3's mean prediction error and storage requirements were lower than those of the

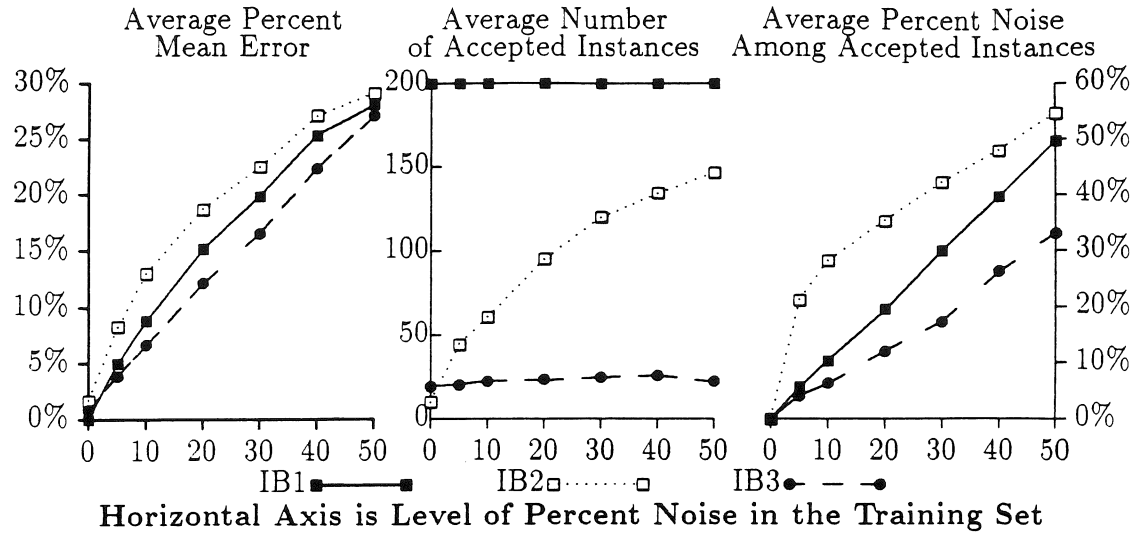


Figure 4.21: IB3's error rate degrades more slowly with increasing noise levels. Its storage requirements and percentage of noise among its accepted instances is also lowest among the three IBL algorithms (target function: $1/(1+e^{x/2})$).

other two algorithms for all non-zero noise levels. IB3's performance was superior in this application because it used a lower percentage of noisy instances in its prediction decisions.

Summary of Empirical Studies

IB3's solid performance in the sigmoid function application does not guarantee that it will perform well in other numeric-prediction applications. Therefore, IB3 was evaluated on the same set of applications that IB1, IB2, and the average guess algorithm were evaluated on in Section 4.2.3. The results are listed in Table 4.8, which is an extension of Table 4.5 on page 73. This table shows that IB3's learning behavior, measured in dimensions of predictive accuracy and storage requirements (i.e., number of acceptable instances), is comparatively consistent with respect to IB2's learning behavior in these applications. IB3 recorded lower average relative errors than IB2 in five of the six applications. Also, IB2's storage requirements were at least three times that of IB3's in these same five applications. IB3's error rates were also at least as low as IB1's in these five applications. IB3's large improvement in IB1's predictive accuracy for the Hungarian data base application was, as expected, similar to its improvement in the equivalent symbolic prediction task as described in Section 4.3.3.

However, IB3 recorded a higher average relative error than its predecessor IBL algorithms for the automobile horsepower prediction task. Its reduction in storage requirements, relative to IB2's, was also smaller for this application. This suggests that the horsepower target

Table 4.8: Average relative error ± standard error and percent storage requirements. IB1 always stored 100% of the training instances.

Target Function	IB1	IB2	IB3	Ave Guess
Hungarian	0.39±0.022	0.43±0.026 36.7	0.23±0.013 6.3	0.46±0.022
Cleveland	0.19±0.005	0.20±0.005 52.9	0.19±0.005 14.2	0.25±0.003
Horsepower	0.04±0.002	0.06±0.003 24.9	0.08±0.004 16.8	0.14±0.006
Number of Months	0.20±0.006	0.22±0.006 57.4	0.20±0.005 18.4	0.24±0.004
Cholesterol Level	0.14±0.007	0.15±0.007 52.3	0.13±0.007 13.6	0.11±0.005
Tumor Size	0.22±0.004	0.23±0.004 62.0	0.21±0.003 16.1	0.17±0.003

function is different than the other five applications. In fact, it is distinguished in at least two ways that affect IB3's behavior. First, there are many more predictor attributes (25) in this application than are used in the other applications. I will present evidence in Section 4.3.4 that, at least for symbolic prediction tasks, IB3's comparative gains in predictive accuracy diminishes (if not vanishes) when the application involves several irrelevant attributes. It is highly probable that some of the 25 predictor attributes are far less relevant for predicting horsepower than others in this task. Second, the average relative errors for the horsepower target function are much smaller than the errors for the other applications. This indicates that the information in the attributes is relatively complete; a strong argument can be made that this application has relatively small amounts of noise. Since the experiments with the sigmoid function showed that IB3's error rate is not lower than the error rates for the other algorithms in non-noisy applications, it is not surprising that IB3 did not outperform the other algorithms in this application.

In summary, IB3 further reduced storage requirements, accepting an average of less than 20% of the instances in each of the applications. Moreover, IB3's average relative error was lower than both IB1's and IB2's in all but one application, which was relatively noise-free. However, its accuracy was better than the accuracy of the simple average guess algorithm in only four of the six applications. Average guess recorded better accuracies than IB3 when the target functions values were normally distributed. It appears that IB3 shares IB1's weakness for this domain characteristic.

Discussion: Other Numeric Prediction Algorithms

These are not the only machine learning algorithms that fare poorly with respect to average guess in some applications. John Gennari (personal communication) also found that, although his CLASSIT algorithm (Gennari, 1990) performed well in comparison to

average guess in the horsepower and three other applications, it also performed comparatively poorly in some of the medical applications used here. In some situations, the average guess algorithm, which was meant to be a "strawman" comparison algorithm, outperforms both the CLASSIT and IBL algorithms.

The CART regression tree algorithm (Breiman *et al.*, 1984) will also perform relatively poorly in some applications. Regression trees are similar to decision trees except that they predict numeric rather than symbolic values; the prediction is the average of the target values of the training instances that were funneled to the leaf chosen to classify the current instance. Also, regression trees choose attribute tests differently, based on their ability to decrease the mean squared difference of the instances' target values in the two subtrees. In a previous study, I empirically compared CART with these IBL algorithms (Aha, 1990). I chose CART because it is an elegant, well-known algorithm for predicting numeric values in the machine learning literature.[7] CART uses the average target value of the training instances that were grouped in each of its leaves as the predicted target value for instances that are indexed to those leaves. In my experiments with CART, Some of the data sets were the same as those used in this present study, although the specific experiments differed. The results showed that the IBL algorithms performed about as well as CART; IB3 had a lower average relative error in half the applications and the difference between their error rates was small. Therefore, CART also performed rather poorly in comparison with the simple average guess algorithm in several applications.

Breiman and his colleagues described case studies in their book that suggested CART performed well in some applications, although no comparisons were made with other numeric prediction algorithms. However, they did note that regression trees have been compared with linear regression in several case studies. They mentioned that CART performed better when the target function was nonlinear and performed worse when the target function was linear. Previous comparative studies have shown that IB1 performed similarly in comparison with linear regression (Kibler, Aha, & Albert, 1989).

I am not aware of any careful domain characterization studies of published machine learning algorithms for numeric prediction that showed when they will outperform such standard methods as linear regression and average guess. There is a great need for these analyses so that users can determine when CART, CLASSIT, and IBL algorithms will display comparatively good predictive behavior. Some arguments have been made that they are more flexible than standard algorithms because they can tolerate missing attributes and process symbolic attribute values. However, extensions of standard statistical algorithms also have these capabilities. CLASSIT and CART's primary contribution appears to be their ability to convey the structure of the target function in a tree-structured hierarchy. The IBL algorithms described in this chapter do not address the issue of comprehensibility. However, they do

[7]To my dismay, I also found the commercial CART software (California Statistical Software, 1984) to be exceedingly difficult to use. This convinced me to avoid using it in this present study.

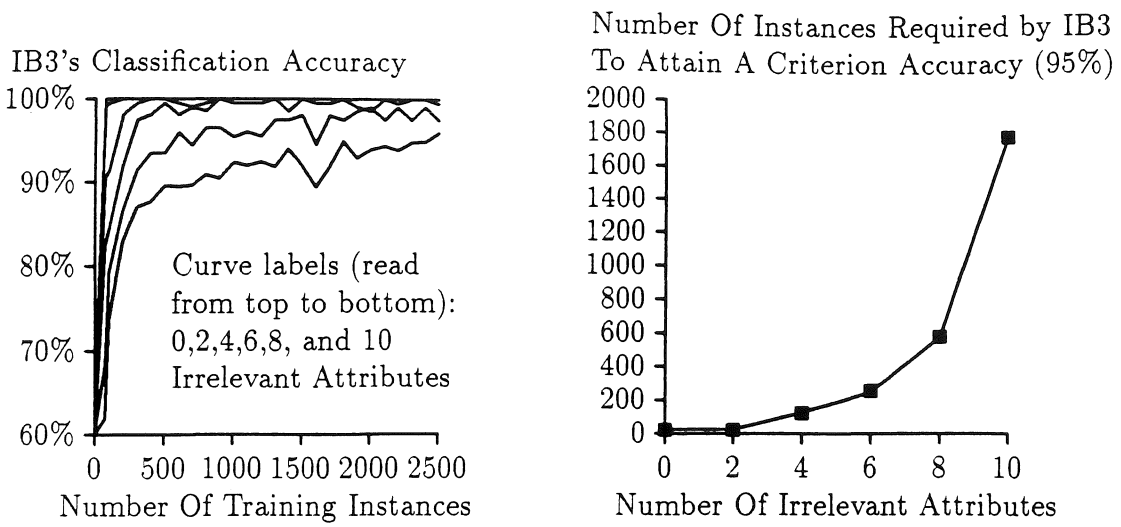


Figure 4.22: IB3's storage requirements (averaged over 25 trials) increase exponentially with the number of irrelevant attributes. All of the attributes are Boolean-valued. Each domain contains only one symbolic target concept and only one attribute is relevant for predicting class membership.

have a contribution; they are comparatively inexpensive to use in an incremental manner. This point is discussed in more detail in Section 8.2.2.

4.3.4 The Effects of Irrelevant Attributes

Although IB3 performed well in the applications described earlier in Section 4.3.3, it has several limitations. Foremost among these limitations is that it does not address the most pressing problem highlighted by the mathematical analyses; it does not attempt to reduce the dimensionality of the instance space. Therefore, it will learn slowly when the dimensionality of the instance space it high.

IB3's flaw can be traced to the design of its primitive similarity function, which incorrectly assumes that all attributes have the same relevance for predicting target values. IB3 could effectively reduce the dimensionality of instance spaces if it could learn to ignore irrelevant attributes. Unfortunately, it does not have this capability.

Figure 4.22 displays a vivid example of this problem. This figure summarizes the results of an experiment where IB3 was applied to a simple binary classification task. All of the predictor attributes were Boolean-valued. Only one of these was relevant for determining class membership. The left-hand graph displays IB3's learning curves when the number of irrelevant attributes was varied. This graph shows that, to reach a criterion accuracy of

95%, IB3's training requirements increase quickly with the number of irrelevant predictor attributes. The graph on the right-hand side of this figure displays this information more clearly. In fact, the number of instances required by IB3 to reach a criterion accuracy goes up exponentially with dimensionality. This observation agrees with the results from the mathematical analysis Chapter 3.

4.4 IB4: Learning Relative Attribute Relevance

This section describes IB4, an extension of IB3 that improves its performance in applications where instances are described by many irrelevant attributes. Results for both symbolic and numeric prediction tasks are described.

IB4 was designed to repair one of the more obvious flaws with IB3, which assumes that the attributes used to describe instances are equally relevant for the purposes of generating accurate target value predictions. IB3's initial assumption seems sound; without further information, there is no reason to believe that some (specified) attributes are more relevant than others for solving prediction tasks. However, IB3 needs to be extended to use feedback from its environment to incrementally learn domain-specific relevance information. IB3 then needs to use this information during classification tasks.

4.4.1 Methods for Learning Attribute Relevance

Algorithms derived from other learning paradigms use different methods to learn attribute relevance. TDIDT algorithms, which induce decision trees using a top-down approach (e.g., Breiman *et al.*, 1984; Quinlan, 1986a; Michie *et al.*, 1984), evaluate each attribute's *localized* relevance (with respect to a pre-specified prediction task) by measuring its information gain for the set of instances located in a specified region of an instance space. Attributes with high information gains are used in the concept description and the other attributes are ignored. Incremental TDIDT algorithms have successfully used this method for solving supervised learning tasks (Utgoff, 1989; Van de Velde, 1990). This evaluation function also served as a good guide for pre-selecting attributes that significantly improved the performance of IBL algorithms (Kibler & Aha, 1987). However, incremental TDIDT algorithms that use this method are currently limited to symbolic-valued attributes. Furthermore, these limited algorithms cannot tolerate noise. Finally, this information is expensive to compute if done in a non-incremental manner and it is not obvious how it can be efficiently computed when the set of accepted instances changes with each prediction attempt. Therefore, this method is not used for IB4, although it may prove to be useful once these problems are solved.

Classifier systems (Holland, 1986; Wilson, 1987) learn attribute relevance in a manner more similar to the method employed by IB4 in that they incrementally receive feedback concerning predictive accuracy and use it to reward parts of the concept description that lead to accurate predictions. In this case, rewards are given to rules that correspond to the region of the instance space that contains the current training instance. Highly rewarded rules are copied and modified while less rewarded rules are eliminated. Attributes that are irrelevant will tend to have their values generalized in rules to "don't care" values. An analogous process could be implemented in IBL algorithms whereby instances could be associated with strengths and reinforcement could lead to the introduction of perturbations of strong instances. However, this process is antithetical to the IBL approach, which states that instances should not be generalized but rather used differently for different purposes. That is, IBL algorithms that learn attribute relevance should update the instance-interpretation process rather than the instances themselves. Michalski (1990) used similar arguments to motivate the need for concept representations that support multiple interpretations.

The AQ class of rule-based systems (e.g., Michalski *et al.*, 1986; Clark & Niblett, 1989) use yet another approach for identifying relevant attributes. These algorithms build maximally-general rules corresponding to areas in instance space populated by subsets of training instances. Rules are built using a general-to-specific search through the space of attributes. This process retains in a rule's conditions references to the minimal set of attributes required to correctly categorize the corresponding set of instances. Although this method focuses prediction on a subset of the attributes that are presumed to be relevant for a given region of instance space, it is non-incremental. Furthermore, it is not obvious that efficient incremental variants exist for this approach since, unlike in decision trees, there is no simple representation modification process analogous to "pulling up" attributes in a decision tree. Therefore, it is unlikely that this method for locating relevant attributes is useful for incremental IBL algorithms.

However, there is an incremental exemplar-based learning algorithm that builds rules of this form (Salzberg, 1988), but it neither assigns attribute relevance based on the attributes used in rule conditions nor perform a general-to-specific search for rules. Instead, Salzberg's NGE algorithm learns estimates of attribute relevance that correspond to the entire instance space (rather than rule-localized relevance information). This information is in the form of attribute weight settings. These weights are similar to the those used in connectionist learning systems (Rumelhart *et al.*, 1987) in that they are attached directly to attributes. However, NGE updates the weights' settings differently. While weights in a connectionist network are updated by propagating rewards through a network, NGE updates its weights by explicitly comparing the attribute values of two instances (i.e., a novel instance and previously processed instance that was used to generate a target value prediction).

IB4's method for representing and updating attribute relevance information was inspired by the methods used in NGE. Dennis Kibler and I reviewed several methods for learning

and representing relevance information that learned useful weight settings but was computationally expensive. As a first step, NGE's suggestion for using attribute weights is quite sound. However, its algorithm for updating weights can be improved. For example, Salzberg (1988) noted that NGE's weight-learning algorithm should reduce the rate of modification change to attribute weight settings over time. Also, this representation assumed that an attribute's relative relevance was invariant across target concepts. IB4's weight-learning algorithm includes solutions to these problems.

4.4.2 Description of the IB4 Algorithm

Table 4.9 details IB4's memory updating algorithm. IB4 is an extension of IB3 that learns attribute relevance information, represented by attribute weight settings. Therefore, IB4's training algorithm is identical to IB3's with two exceptions. First, IB4's similarity functions uses this attribute relevance information so that more relevant predictor attributes are emphasized (i.e., their relative Attribute_difference range is stretched) and less relevant predictor attributes are de-emphasized (i.e., their relative Attribute_difference range is shrunk). Second, IB4's learning algorithm extends IB3's to update these attribute weights settings. IB4 uses a nonparametric reinforcement learning strategy to assign higher weight settings to attributes that assist in correct classifications and lower settings to other attributes. IB4 effectively transforms the initial instance space so that attributes with high relevance have larger ranges of values. IB4's testing algorithm is identical to IB3's except for its modified similarity function.

IB4's similarity function is defined as

$$\text{Similarity}(x, y, t, P) = \frac{1}{\sqrt{\sum_{i \in P} w_{t_i} \times \text{Attribute\_difference}(x_i, y_i)}}, \tag{4.4}$$

where w_{t_i} is predictor attribute i's weight when predictions are requested for target concept t. A unique set of attribute weights is learned for each target concept. IB4 has great flexibility; it assigns an attribute's relevance based on the prediction task. Therefore, its similarity definition is task-dependent. This capability is not shared by IB4's predecessors but is shared by other learning algorithms (e.g., decision-tree, rule-based) that use different attributes to classify instances dependent on the target concept. For example, the similarity of a tiger and a cat is higher if the task is to predict whether they are animals than whether they are potential pets.

IB4's prediction function is similar to IB3's in that only acceptable instances are used to generate target value predictions. However, IB4 maintains a separate set of attribute weight settings and partial concept description for each target concept (i.e., each different target attribute value). This endows IB4 with yet another ability not shared with its predecessors: IB4 does not assume that concepts are disjoint and exhaustive. Instead, instances can be

Table 4.9: IB4's definition for its memory updating function.

Key: T: Training set
P: Set of predictor attributes
t: The target attribute
k: Number of most similar instances used
α & δ: Confidence thresholds for acceptance and dropping
PCD: Partial concept description

Train(T, P, t, k,tolerance_threshold,α,δ)
1. Set global variables
2. for each $x_i \in T$ do
   2.1 $x_i \leftarrow$ Pre_process(x_i, P)
   2.2 prediction $\leftarrow$ Performance(x_i, P, t, k,α)
   2.3 Learn(x_i,prediction,t,tolerance_threshold,δ)

Performance(x, P, t, k, α)
1. $S \leftarrow \emptyset$
2. $\forall y_i \in$ PCD: $S \leftarrow S \cup \{\langle y_i, \text{Similarity}(x, y_i, t, P)\rangle\}$
3. KSET $\leftarrow$ k_most_similar_acceptable_instances(S, k, α)
4. return Target_value_prediction(KSET,t, k)

Learn(x,prediction,t,tolerance_threshold,δ)
1. if ($|x_t -$ prediction$| >$ tolerance_threshold)
   then PCD $\leftarrow$ PCD $\cup \{x\}$
2. $S \leftarrow$ set of stored instances at least as similar as
   the k^{th} most similar acceptable instance
3. Update_attribute_weights(S, x,prediction,t)
4. Update_prediction_records(S)
5. Discard_significantly_poor_instances(S, δ)

predicted to have multiple, single, or no target values. Aha (1989b) described experiments that showed how IB4 can learn overlapping concepts.

This strategy requires additional work because an instance is classified with respect to n concept descriptions when a target has n potential values. Whenever an instance is misclassified, it is added to the corresponding target value's partial concept description. Since IB4 represents concepts independently of each other, instances are either members ("positive") or non-members ("negative") of a concept. Therefore, each target concept's description groups all non-members together, independent of their other classifications.

IB4's prediction that an instance x has target value v for some target attribute t is decided by a vote among the k most similar acceptable instances in t's partial concept description for target value v. One prediction is made for each potential target value. However, many prediction tasks assume that concepts are disjoint and exhaustive. This is true for the seven applications used previously to evaluate algorithms IB1, IB2, and IB3. IB4 must act differently during testing to exploit this information. Therefore, I modified IB4's prediction function so that, when a target attribute has more than two possible values, IB4's estimate of membership is defined as a function of the most similar acceptable instance in v's partial concept description that is a concept member and the most similar acceptable that is *not* a concept member. The intuition is that an instance's estimate of membership is defined to increase with its similarity to the most similar acceptable instance with target value v and decreases with its similarity to the most similar acceptable instance with a different target value. This estimate of membership function is defined as

$$\text{Estimate\_of\_membership}(x, t, v, P) = \frac{1/\text{Similarity}(x, neg, t, P)}{1/\text{Similarity}(x, neg, t, P) + 1/\text{Similarity}(x, pos, t, P)}$$

where *pos* is a nearest acceptable neighbor of x whose target value is v and *neg* is a nearest acceptable neighbor of x that has a different (i.e., negative) target value.[8] The target value with the highest estimate of membership is then selected as IB4's prediction for the test instance's target value. (Ties are randomly broken among the target values with the highest estimate.)

IB4's memory updating function is an extension of IB3's. In addition to maintaining prediction records for each stored instance, IB4 also learns each target concept's attribute weights through another performance feedback process. Attribute weights are initialized to be equal, vary in the range $[0, 1]$, and are normalized to sum to one. Each weight is updated after each prediction attempt during the training process. Weights are derived from two other variables as follows: (for attribute i and some target concept t)

$$\text{Weight}_{t_i} = max(\frac{\text{CumulativeWeight}_{t_i}}{\text{WeightNormalizer}_{t_i}} - 0.5, 0) \tag{4.5}$$

All attribute weights are updated after each training instance x is classified. The k most similar acceptable instances in the partial concept description are used to update the weights, as described in Table 4.10. The CumulativeWeight numerator is incremented by a fraction of the value added to the WeightNormalizer denominator. Attribute weights are increased when they correctly predict classifications and are otherwise decreased. That is, the CumulativeWeight's increment is high when its attribute's value assists making a correct classification decision and is otherwise low. An attribute's pair of values assists in making

[8]Recall that instances are either members or non-members of a given concept. Therefore each concept description groups all non-member instances together, independent of their other concept memberships.

Table 4.10: IB4's weight-updating algorithm.

Variable Key: x – instance being classified
y – one of the k most similar acceptables
t – the target concept
Λ – the higher observed frequency among t_v's actual and predicted class members
λ – the observed frequency of y's membership status in t_v

for each attribute i:

1. let Difference = $|x_i - y_i|$
2. if (x's classification was correctly predicted)
   then
   2.1a $\text{CumulativeWeight}_{t_i} = \text{CumulativeWeight}_{t_i} + (1\text{-}\lambda)\times(1\text{-Difference})$
   2.2a $\text{WeightNormalizer}_{t_i} = \text{WeightNormalizer}_{t_i} + (1\text{-}\lambda)$
   else
   2.1b $\text{CumulativeWeight}_{t_i} = \text{CumulativeWeight}_{t_i} + (1\text{-}\Lambda)\times\text{Difference}$
   2.2b $\text{WeightNormalizer}_{t_i} = \text{WeightNormalizer}_{t_i} + (1\text{-}\Lambda)$

correct classifications when either (1) a correct classification occurs and the two instances' attribute values are similar to each other or (2) an incorrect classification occurs and they are dissimilar. Otherwise, the CumulativeWeight's increment is small.

IB4's weight updating algorithm, like IB3's significance test, attends to classes with low observed relative frequency to tolerate skewed concept frequency distributions. The variable Λ refers to the higher frequency for instances in the concept distribution (which are labeled as either "positive" or "negative"). Variable λ refers to this same frequency for the saved instance's target value, which may either be the lower or the higher of the two frequencies. Matches of instances of the less frequent value yield larger adjustments of attribute weights. Otherwise, the more frequent value's instances would have an overwhelming affect on the settings of the weights.

The best way to understand this weight-learning algorithm is to work through a detailed example. Suppose the target concept is "Ph.D. student" and the instances (people) are described with three Boolean attributes ("is enrolled","has M.S. degree", and "is married"). Suppose also that IB4 has been trained on four instances, only one of which was a Ph.D. student (with attribute values <True,True,True>), the resulting CumulativeWeights settings are (0.65,0.65,0.65), the WeightNormalizers are all 0.75, and $k = 1$. Finally, assume for the moment that this positive instance is acceptable throughout this example. These settings and the derived attribute weights are listed in the first two rows of Table 4.11 The next three lines in this table describe (1) which attributes' values match between the fifth training instance and its most similar acceptable instance, (2) the resulting CumulativeWeight and WeightNormalizer settings, and (3) the attribute weight settings after the fifth training instance has been processed. If the fifth instance is incorrectly classified as a Ph.D.

Table 4.11: Intermediate results of IB4's weight-learning algorithm.

Prediction Result	Membership Frequency	Attributes		
		Enrolled	Degree	Married
	0.25	0.65/0.75 $0.\bar{3}$	0.65/0.75 $0.\bar{3}$	0.65/0.75 $0.\bar{3}$
wrong	0.2	mismatch 0.90/1.00 0.42	mismatch 0.90/1.00 0.42	match 0.65/1.00 0.16
correct	$0.\bar{3}$	match 1.70/1.80 1.0	mismatch 0.90/1.80 0.0	mismatch 0.65/1.80 0.0

student and has attribute values <False,False,True> (i.e., not enrolled, no M.S., married), then the new CumulativeWeight settings are (0.9,0.9,0.65) and the WeightNormalizers all become 1.0. Finally, the last three lines describe the situation after the sixth training instance was processed. If the sixth instance is correctly classified as a Ph.D. student and has attribute values <True,False,False> (i.e., enrolled, no M.S., not married), then the new CumulativeWeight settings are (1.7,0.9,0.65) and the new WeightNormalizers are 1.8. This indicates that IB4 has learned that the attribute "is enrolled" is more predictive of the Ph.D. student class than either "has M.S. degree" or "is married" for these training instances. This weight-learning algorithm assigns higher weight settings to predictive attributes than to less relevant attributes.

In summary, IB4 is an extension of IB3 that does *not* assume all attributes have equal predictive relevance. Instead, it incrementally learns settings for attribute weights that denote relative attribute relevance. These weights are then used in IB4's similarity function. Updating the attribute weights after each classification attempt continuously changes a concept's similarity function. Thus IB4 learns domain-specific similarity functions independently for each target concept.

4.4.3 Empirical Results with IB4

Symbolic Prediction Tasks

IB4 can outperform IB3 in many applications whose instances are described by multiple irrelevant attributes. For example, Figure 4.23 shows an extension of the right-hand graph in Figure 4.22 on page 98. Although IB3's storage requirements for attaining criterion (i.e., 95% classification accuracy) increase exponentially with the number of irrelevant attributes, IB4's storage requirements remain relatively the same. IB4's learning rate will not decrease

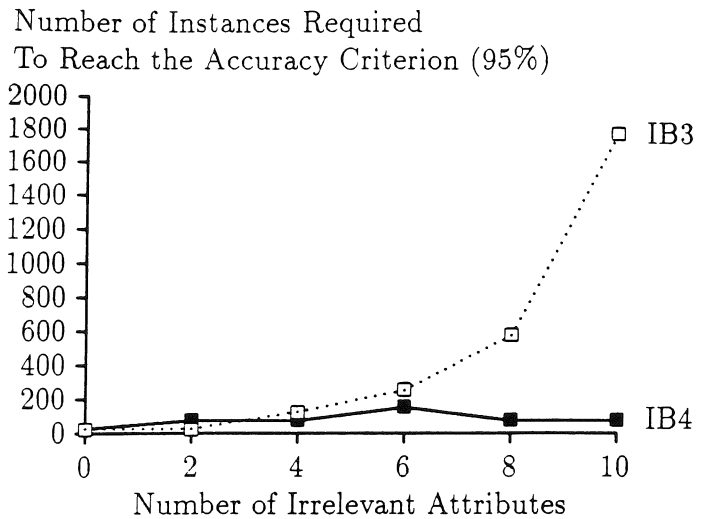


Figure 4.23: IB4's storage requirements (averaged over 25 trials) for achieving a criterion classification accuracy increase more slowly than IB3's with the number of irrelevant attributes.

as quickly as will IB3's when applied to some domains whose attributes vary greatly in their relevance for generating predictions.

Of course, IB4's benefits on this example may be restricted to this application domain. Therefore, IB4 was evaluated on the set of applications described in Table 4.2 on page 65. The results are summarized in Table 4.12 alongside the results obtained for the other algorithms tested and are discussed in detail in the following paragraphs. IB4 achieved higher classification accuracies than the other IBL algorithms when the application's instances were described by numerous irrelevant attributes. However, IB4 performed poorly when trained on small numbers of instances. IB4's storage requirements are defined as the number of unique training instances that it accepted among its set of partial concept descriptions.

Noisy Artificial Domains:

IB4 performed terribly on the LED-7 domain. It's average accuracy after 200 training instances was 32.1%, which is about 40% lower than the accuracies recorded by the other algorithms. The main reason that IB4 learns slowly here is that few (about 10) positive non-noisy instances of each concept are expected to exist in a training set of 200 instances. This isn't a problem for the other IBL algorithms because they assume that the concepts are disjoint and exhaustive. However, IB4 doesn't make this assumption and learns a partial concept description for each target value. Since it does not benefit from the knowledge that an instance must be a member of exactly one concept, IB4 requires more instances to learn concepts when this information is useful. However, it eventually achieves reasonably high classification accuracies. Figure 4.24 shows IB4's average learning curve for this domain over

Table 4.12: Average % accuracy ± standard error and % storage requirements (lower lines).

Database	IB1	IB2	IB3	IB4	C4
LED-7	71.6±0.4	63.0±0.9 41.6	72.5±0.4 20.1	32.1±3.0 21.3	68.9±0.5
Waveform-21	75.5±1.1	68.4±1.1 32.3	74.2±1.2 11.1	76.9±1.0 13.4	71.5±1.2
Cleveland	75.1±0.8	71.4±0.9 32.0	78.4±0.9 3.9	78.1±0.7 4.2	75.2±1.2
Hungarian	56.1±2.2	53.1±2.4 36.9	79.4±0.9 4.3	79.8±0.7 3.9	77.6±0.9
Voting	91.8±0.4	90.9±0.5 11.6	90.6±0.6 3.5	93.8±0.4 3.0	96.1±0.6
LED-24	47.9±0.6	43.7±0.8 60.1	46.6±0.7 25.3	66.1±0.6 25.4	66.9±2.1
Waveform-40	68.6±0.7	64.0±0.7 38.3	67.2±1.1 11.8	72.1±1.2 12.6	70.9±1.0

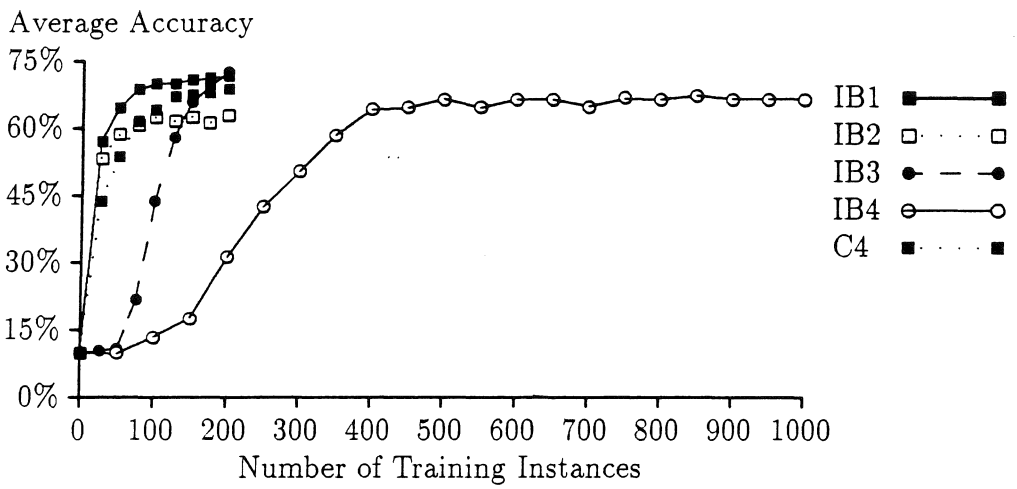


Figure 4.24: Learning curves for the LED-7 application.

larger-sized training sets; it is an extension of Figure 4.17 on page 90. IB4's classification accuracy begins to level off and approximate the other algorithm's accuracies after 500 training instances.

IB4 recorded the highest classification accuracy among the algorithms tested on the Waveform-21 noisy artificial domain (CART's accuracy was 70% (Breiman *et al.*, 1984)). This occurred because the domain's instances were described by many irrelevant attributes

(i.e., recall that CART and C4 learned accurate decision trees that used few attributes). IB4's storage requirements were also low, almost matching IB3's.

Although the accuracies recorded from the experiment with the LED-7 domain approximated the Bayes optimal rate (74%), none of the algorithms approached this rate for the Waveform-21 domain (86%). This leaves open to question why the biases of these learning algorithms more easily match the characteristics of the LED-7 domain. Both the IBL and decision tree algorithms required that similar instances have similar classifications to ensure high classification accuracies. It's probable that the Waveform-41 domain is highly disjunctive, which would cause these algorithms to perform sub-optimally. Additional analyses of this domain would be required to evaluate this possibility.

Databases with Imperfect Attribute Sets:

IB4 performed well in the experiments with the cardiology databases; its classification accuracies and storage requirements were similar to IB3's. It also had higher classification accuracies than IB1 and C4 on both data sets. This is typical of IB4's behavior when the application involves few target concepts and the instances are described by relatively small numbers of attributes, some of which are irrelevant.

Noise-Free Database:

The instances contained in the Congressional Voting database are described by many attributes that can be safely ignored without sacrificing classification accuracy. Therefore, it is not surprising that IB4's accuracy on this domain is superior to that of the other IBL algorithms. In some sense, IB4 corrects IB3's bias to expect noise by performing simple transformations on the instance space (i.e., it stretches and shrinks the predictor attribute dimensions) that yield simpler spaces in which higher classification accuracies can be obtained. IB4's storage requirements were also lower than IB3's in this application. However, IB4's accuracy is still lower than C4's in this application. The latter algorithm's learning bias for this application is clearly superior than the IBL algorithm's.

Domains with Irrelevant Attributes:

IB4's benefits were most apparent in the experiments with the LED-24 and Waveform-40 domains, which contain 17 and 19 additional irrelevant attributes respectively. The algorithms' average learning curves for the LED-24 attribute domain are displayed in Figure 4.25. As mentioned earlier, IB4's initially slow learning curve is due to its lack of information that the concepts are disjoint and exhaustive in these domains. However, IB4's learning curve eventually rises above the other algorithms' curves. After the first 300 training instances, IB4's average learning curve is significantly higher than IB1's ($t(13) = 7.20, p << 0.0005$). Likewise, after the first 200 training instances, IB4's average learning curve for the Waveform-40 domain is significantly higher than IB1's ($t(3) = 7.96, p < 0.0025$). IB4 performs well in

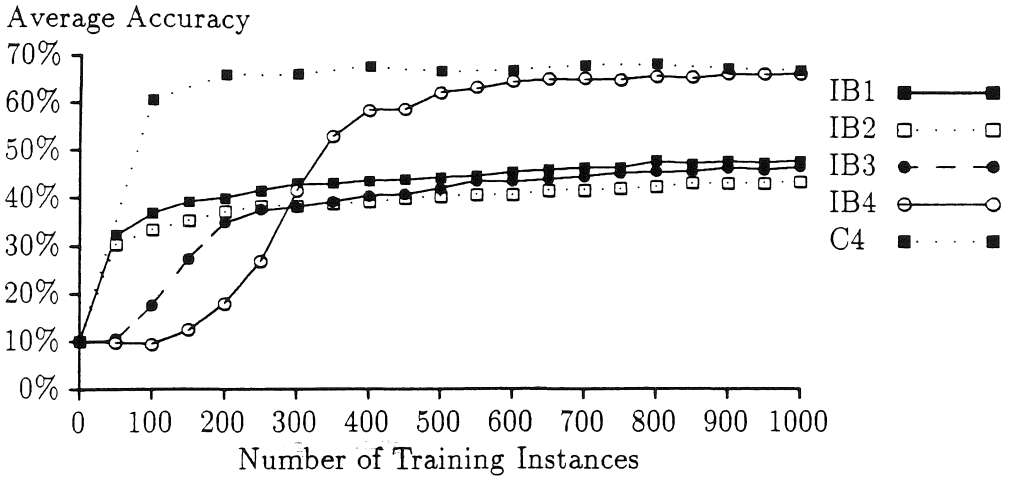


Figure 4.25: Learning curves for the LED-24 application.

these applications, recording accuracies similar to C4's, by learning the relative relevance of the predictor attributes for generating accurate predictions.

Numeric Prediction Tasks

IB4 was also applied to the six numeric-prediction tasks used in Sections 4.2 and 4.3. Its weight-learning algorithm is slightly different for numeric prediction tasks. In this case, the frequency of an instance's class is replaced by the observed frequency that training instance's target values are in the same target interval. Similarly, the frequency for instances with different classifications is replaced by the frequency of instances whose target values lie in different intervals. Furthermore, I have not yet investigated IB4's learning behavior for learning separate sub-intervals of the target function. Although it would be possible to study IB4's behavior in this mode, it would be difficult because the set of target function intervals change each time a new maximum or minimum target function value is processed during training. Therefore, IB4 maintains a single partial concept description for numeric prediction tasks.

The results of the six applications are summarized in Table 4.13. In general, IB4's behavior is highly similar to IB3's for these applications. IB4's error rates were lower in for four and its storage requirements were lower for three of the six applications. However, these differences were small. The reasons for this is that none of these applications' instances are described by numerous irrelevant attributes, which have been removed during the process of selecting attributes for describing instances in these databases. IB4 should record faster learning rates in applications with numerous irrelevant attributes.

Table 4.13: Average relative error ± standard error results. Percent average storage requirements are shown on the second lines.

Target Fn	IB1	IB2	IB3	IB4	Ave Guess
Hungarian	0.39±0.022 100%	0.43±0.026 36.7%	0.23±0.013 6.3%	0.20±0.008 7.5%	0.46±0.022
Cleveland	0.19±0.005 100%	0.20±0.005 52.9%	0.19±0.005 14.2%	0.18±0.005 13.9%	0.25±0.003
Horsepower	0.04±0.002 100%	0.06±0.003 24.9%	0.08±0.004 16.8%	0.07±0.003 17.8%	0.14±0.006
Number of Months	0.20±0.006 100%	0.22±0.006 57.4%	0.20±0.005 18.4%	0.20±0.007 17.3%	0.24±0.004
Cholesterol Level	0.14±0.007 100%	0.15±0.007 52.3%	0.13±0.007 13.6%	0.14±0.007 13.3%	0.11±0.005
Tumor Size	0.22±0.004 100%	0.23±0.004 62.0%	0.21±0.003 16.1%	0.21±0.003 17.5%	0.17±0.003

Table 4.14: Average relative error ± standard error results and average percent storage requirements. The second column lists the results for the original horsepower function while the second list the results when 20 additional irrelevant attributes were used.

Algorithm	Original Function	Modified Function
IB1	0.0434±0.002 100.0	0.0881±0.004 100.0
IB2	0.0593±0.003 24.9	0.1012±0.002 35.7
IB3	0.0764±0.004 16.8	0.1074±0.004 18.9
IB4	0.0697±0.003 16.0	0.0785±0.004 17.8

To test this claim, I applied these IBL algorithms to a variant of the horsepower target function application, whose instances were described by twenty additional irrelevant Boolean-valued attributes. The values of these additional attributes were randomly determined. The results, summarized in Table 4.14, show that IB4 recorded lower error rates and lower storage requirements than the other three algorithms. IB4's increase in predictive accuracy after 144 training instances is significant in comparison with each of the other algorithms (IB1: $t(24) = 1.32, p < 0.1$, IB2: $t(24) = 2.00, p < 0.05$, IB3: $t(24) = 1.76, p < 0.05$). Their average learning (error) curves are shown in Figure 4.26. After processing 50 training instances, IB4's average learning curve was significantly lower than the other algorithms' curves (IB1: $t(4) = 4.76, p < 0.005$, IB2: $t(4) = 4.49, p < 0.01$, IB3: $t(4) = 9.28, p < 0.0005$). Therefore, IB4 can significantly increase predictive accuracy and learning rate in some numeric prediction tasks that contain large numbers of irrelevant attributes.

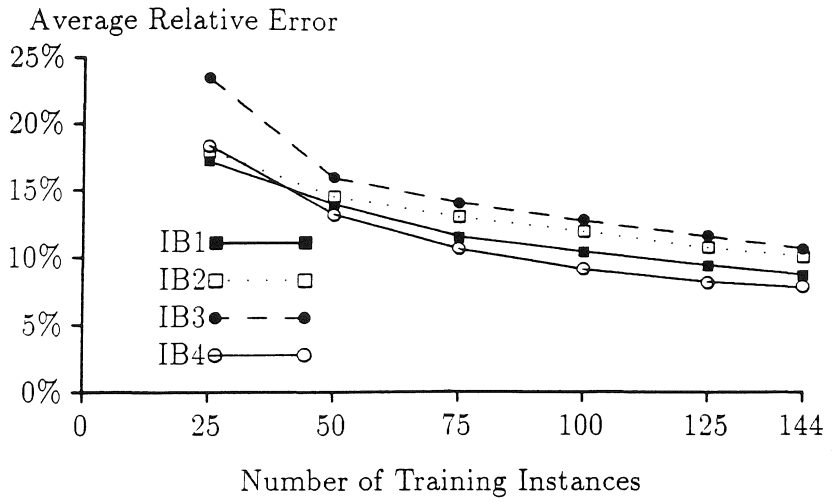


Figure 4.26: Learning curves for the horsepower plus 20 irrelevant attributes target function.

4.5 Discussion

IB4 has many flaws and is not a polished algorithm. Rather, it is a first step towards solving one of the most critical problems affecting the instance-based approach, commonly known as the "curse of dimensionality." This problem was anticipated by the mathematical analyses in Chapter 3, where it was proved that a concept's boundary area grows exponentially in size with an increased number of predictor attributes. Therefore, an exponentially larger number of training instances is required to yield accurate concept descriptions when the dimensionality is increased.

The experiments have shown that IB4's learning rate is initially slow. However, in several cases IB4's learning curve rises far above those for the other IBL algorithms. Fortunately, the behaviors of the other algorithms suggest several ways in which IB4 can be improved. Considering how slow both IB3 and IB4 are, it would seem prudent to allow these algorithms to behave like IB1 during the early stages of the training process. One way to accomplish this is to save more of the training instances by relaxing the error threshold.

However, this will not necessarily result with higher numbers of acceptable instances. Therefore, the confidence level parameter for accepting instances also needs to be relaxed, but should *later* be reset to a higher (tighter) value to reduce the number of noisy instances used for generating predictions. The confidence level parameter for dropping instances should also follow this pattern to ensure that instances aren't dropped prematurely but are dropped after proving themselves to be poor classifiers. That is, the correct setting for these parameters should be *learned.* One demonstration where this method might prove effective is in application to learning the exclusive-or function when 25 additional irrelevant attributes are

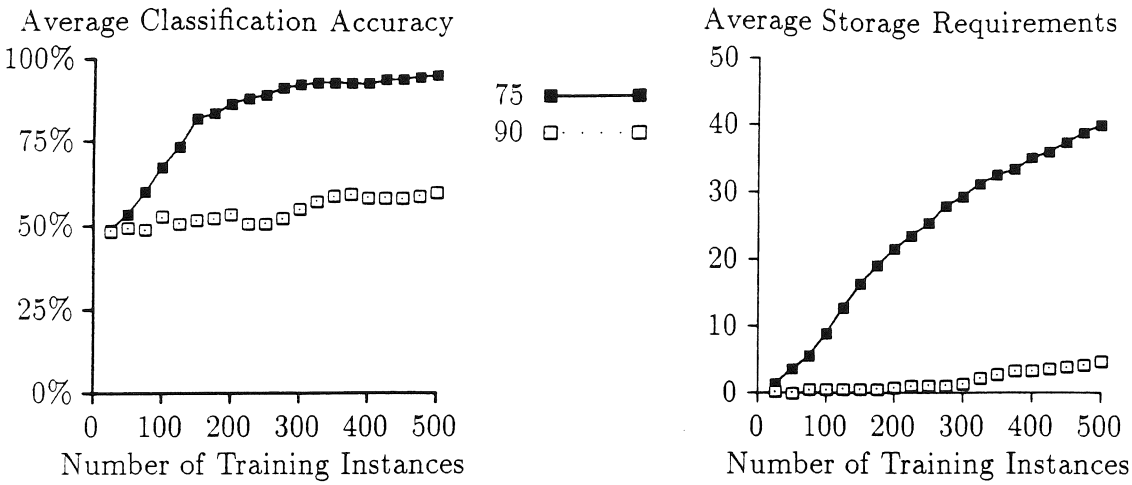


Figure 4.27: IB4 learns more quickly with a lower setting for its confidence level for acceptance parameter when applied to the exclusive-or function in the presence of 25 irrelevant attributes.

used to describe each instance. IB4's learning behavior will vary greatly depending on the setting of the confidence level for acceptance parameter. In this case, Figure 4.27 shows that IB4 performs much better with a 75% setting than with a 90% setting. This is because the higher setting makes it more difficult for instances to become acceptable; too few instances were accepted by IB4 with the higher parameter value setting. In fact, no instances were accepted in some of the 25 learning trials with the 90% setting there were 100 test instances per trial). Methods for learning the settings of IB3's parameters is a fertile avenue for future research of instance-based learning algorithms.

In the previous experiments, it would seem that IB4's learning rate cannot compete with C4's. However, this is not completely true. If C4 is trained on applications in which it must learn a relatively large tree and many attributes are relevant, then IB4's learning curve could reach an asymptote sooner than C4's. This occurred in experiments with variants of the LED-24 domain. The four IBL algorithms and C4 were applied to two variants of this domain in which the relevant attributes were randomly selected for each concept. That is, instead of using the same set of seven of the 24 attributes for describing the seven relevants, a different set was chosen for each of the digits. This choice was randomly selected and different for each of the 25 learning trials. The first experiment corresponds to the situation when no noise was added to the data (other than the 17 irrelevant attributes) while instances in the second experiment were subjected to a 5% noise probability. Table 4.15 summarizes the average results. As expected, IB4 outperformed the other algorithms in all four applications, both in terms of higher classification accuracies and lower storage requirements. However, IB4's average classification accuracy seems *disturbingly* higher than C4's. In fact, there is a simple reason why IB4 seemingly outperformed C4 here. C4's accuracy was lower because

Table 4.15: Average % classification accuracies and storage requirements on 2 variants of the LED-24 domain. IB4 performs well even when concepts don't share relevant attributes.

Algorithm	No Noise		5 Percent Noise	
	Accuracy	Storage	Accuracy	Storage
IB1	78.8%	1000	62.4%	1000
IB2	71.6%	316.3	58.0%	447.9
IB3	74.5%	275.0	62.4%	276.7
IB4	**94.2%**	**255.9**	**80.1%**	**247.5**
C4	87.7%		69.6%	

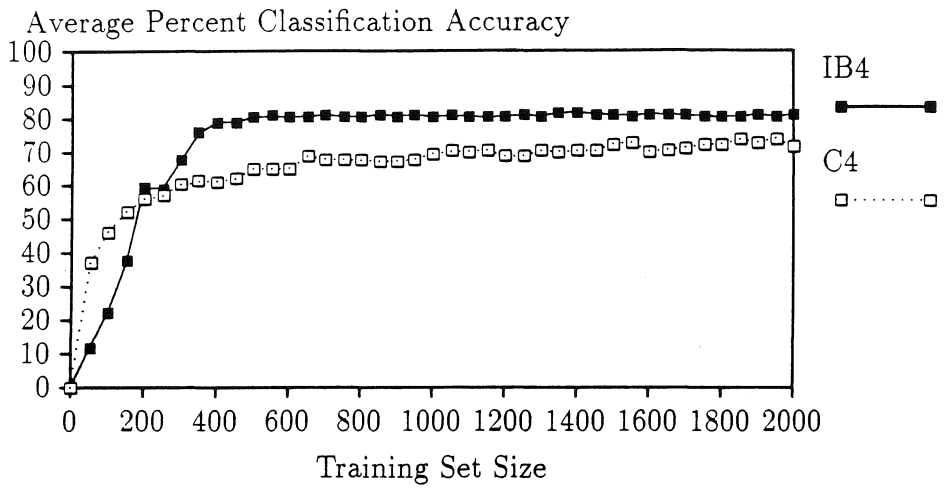


Figure 4.28: IB4 learned more quickly than C4, not more accurately, when the relevant attributes were randomly selected for each concept.

the decision tree algorithm partitioned the training instances into leaves with small numbers of instances, which were subsequently pruned. A learning curve analysis for the second experiment, shows that C4 learns more slowly than IB4, but will probably achieve the same accuracies once these leaves become large enough to be statistically significant. Figure 4.28 shows the average learning curves for IB4 and C4 through training set sizes of 2000 instances. This experiment demonstrated the utility of learning separate concept descriptions for each target concept. For example, C4's average learning curve would be higher if it was directed to create a separate decision tree for each digit, much as IB4 is directed to learn a separate partial concept description for each digit.

IB4's weight-learning algorithm has not been analyzed mathematically and, given that PAC-learning algorithms are difficult to generate for IB3 (Volper, personal communication), it is difficult to specify precisely when IB4 will perform well. However, the case studies

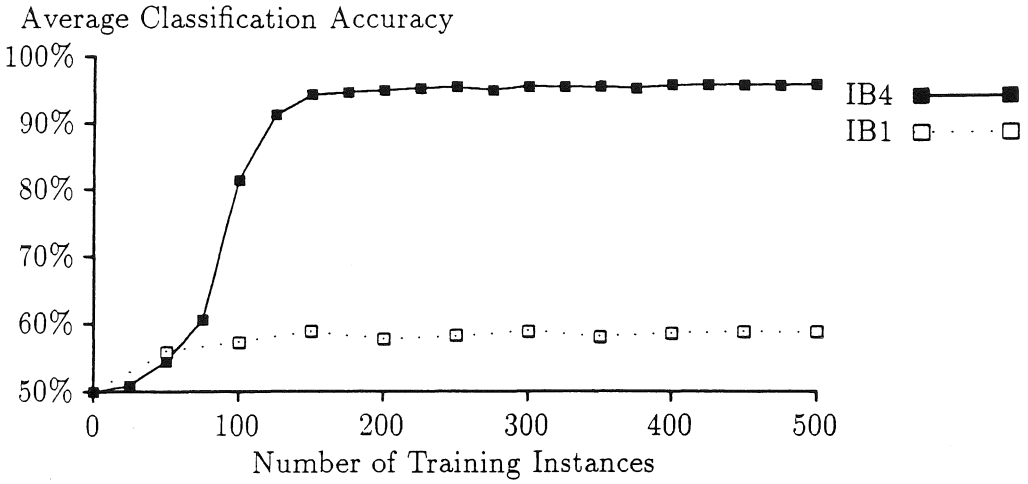


Figure 4.29: IB4 can learn which of 251 predictor attributes is relevant.

described in this section provided some clues. IB4 outperformed the previous IBL algorithms in applications where the instances are described by numerous irrelevant attributes. IB4 will perform well in idealized applications such as the one described in Figure 4.29, where the target concept is defined by a single relevant and 250 irrelevant attributes (the curves are averaged over 25 trials). IB4 can also learn graded and overlapping concepts (Aha, 1989b). However, it will learn more slowly than decision tree algorithms when irrelevant attributes abound and few attributes are relevant. Also, its initial learning rate is easily the slowest of the four IBL algorithms tested in this chapter. Finally, IB4's capability for predicting numeric values has not yet been analyzed. Chapter 5 highlights several other problems with IB4 (and the other algorithms) and discusses initial attempts to solve them.

4.6 Chapter Summary

This task-oriented chapter contained empirical analyses of three IBL algorithms for addressing three problems with the instance-based learning approach: excessive storage requirements, sensitivity to noise, and sensitivity to irrelevant attributes. IB2 was developed to reduce IB1's storage requirements. The only instances saved by IB2 are those that lie in or nearby regions of the instance space where the derivative of the target function is high. Although IB2 significantly reduces storage requirements, it also increases IB1's sensitivity to noise. This lead to the development of IB3, a noise-tolerant extension of IB2 that employs a confidence intervals of proportions test to ensure that only instances with significantly good classification accuracy are allowed to partake in classification decisions. However, these three algorithms do not perform well when instances are described in part by numerous irrelevant

attributes. IB4, an extension of IB3, was developed to solve this problem. It uses performance feedback information from classification outcomes to modify the impact of attributes in subsequent classification decisions. Experiments with IB4 showed that it extends the set of applications in which IBL algorithms can perform reasonably well.

The next chapter reviews commonalities of these four IBL algorithms with respect to the framework described in Chapter 2. Several design decisions were made to simplify these IBL algorithms, thus making them easy to understand, implement, and test. However, this resulted with a set of highly constrained algorithms. Chapter 5 highlights many of these simplifications and suggests alternatives that will lead to improved learning behavior.

Chapter Acknowledgements

Dennis Kibler originally suggested examining the algorithm that became known as IB2. He also suggested using prediction records to track an instance's utility. We originally used an ad-hoc algorithm for locating noisy instances. Based on suggestions from Jim Wogulis, Wayne Iba, and other reviewers that we use a more principled method for IB3's acceptance procedure, I managed to devise the current strategy that incorporates a significance test. Thanks to David Ruby for helping me locate this test. Doug Fisher, Ray Bareiss, Peter Clark, Rob Holte, Ross Quinlan, and several (other!) anonymous reviewers added valuable suggestions for improving the algorithm and its presentation. Steven Salzberg's (1988) weight-learning algorithm provided the inspiration for the one used in IB4. Dennis Volper and George Shackelford have recently made progress towards mathematically analyzing IB3's behavior, but complete analyses remain elusive.

Chapter 5
Parametric Studies

The previous chapter described empirical analyses for three IBL algorithms that address three problems left unsolved by IB1. Although the analyses proved useful for analyzing how well the algorithms fulfilled their intentions, no analyses of the algorithms' component-level design decisions were presented. That is, for each algorithm, a choice was made for instantiating each of the IBL framework's components without analysis of alternatives. Potential alternatives are empirically analyzed in this chapter, which describes a set of five component-level parameter studies.

The analyses in this chapter are partitioned into three sections. Each section corresponds to one of the three components in the IBL framework, which was described earlier in Section 2.2. The following list briefly reviews these components.

1. *Pre-processing:* This component pre-processes the instances given to the performance and learning components. The only pre-processing function used in IB1 through IB4 is the *normalization* function, which linearly normalizes the ranges of numeric-valued attributes.
2. *Performance:* This component is responsible for generating target value predictions given a set of partial concept descriptions. Its two sub-functions are the *similarity* and *prediction* functions.
3. *Learning:* This component includes the *memory updating* function, which determines which instances to include in partial concept descriptions. It also maintains information concerning the predictive behavior of saved instances and predictor attributes.

Alternative definitions for each of these components' functions are empirically investigated in this chapter. The purpose of these studies is to compare different algorithms for implementing component sub-functions and to characterize how they affect the behavior of IBL algorithms. The alternatives selected in these investigations were carefully chosen; they are intuitively appealing and most have been used previously in similar algorithms.

Section 5.1 describes a comparison study of three different methods for normalizing instances. Section 5.2 describes studies of alternative definitions for the performance component functions. It includes two investigations with the similarity function. The first investigation compares three methods for processing missing attribute values while the second

examines three methods for processing symbolic (i.e., categorical) attribute values. This is followed by a study of alternative prediction functions, ones that use more than one instance (i.e., $k > 1$) to generate target value predictions. Finally, Section 5.3 reviews an alternative for the IBL learning component. The extensions of IB1 described in the previous chapter are *instance-filtering* IBL algorithms; they retain a subset of the training instances, filtering many from the target concepts' partial concept descriptions. Section 5.3 contains a study of *instance-averaging* IBL algorithms (e.g., Bradshaw, 1987), which average correctly classified instances with their classifiers.

All of the comparison studies in this chapter almost always evaluate alternative function definitions by testing them on the seven symbolic and six numeric prediction tasks introduced in Chapter 4. The same training and testing methodologies are also used.

These studies have not previously been conducted at the level of detail presented in this chapter. Thus, the results of these investigations allowed me to draw several useful conclusions in Section 5.4. However, this chapter's scope is limited: factorial analyses are not included, which would have required a large number (108) of additional experiments. Therefore, the results may be misleading and, in many cases, require additional testing to eliminate the influences of other potential variables. Nonetheless, the results show that (1) choosing alternative component definitions can significantly affect the IBL algorithms' learning behavior and (2) future research is required to characterize the conditions under which each design alternative will support good learning behavior.

5.1 Alternative Pre-Processing Components

This section describes a comparison study that examines three different methods for normalizing instances. The first method, named the *linear normalization* function, was used in all of the previous experiments and was also used by Salzberg (1988) in his NGE exemplar-based learning algorithm. It normalizes the ranges of all numeric attributes to $[0, 1]$. The motivation for this study is to investigate whether this method is flawed. For example, IB1, IB2, and IB3 may perform better with unnormalized attribute values if the application's attribute ranges correspond to their predictive relevance. In such cases, the similarity function should use the unnormalized values so that the relevant attributes have greater influence on similarity computations. This normalization method may also be insufficient when the numeric predictor attributes' distribution of values contains outliers. Figure 5.1 shows an example of this situation. This is the distribution of the main memory size attribute's values in Ein-Dor and Feldmesser's (1987) database on relative cpu performance. Four of the computers in their database have a 64M main memory size. When the linear normalization method is used, these outliers cause the other attribute values to look more similar. For example, if two of this attribute's values are chosen at random from instances in this database, then there is a high probability that they will have a relatively small normalized

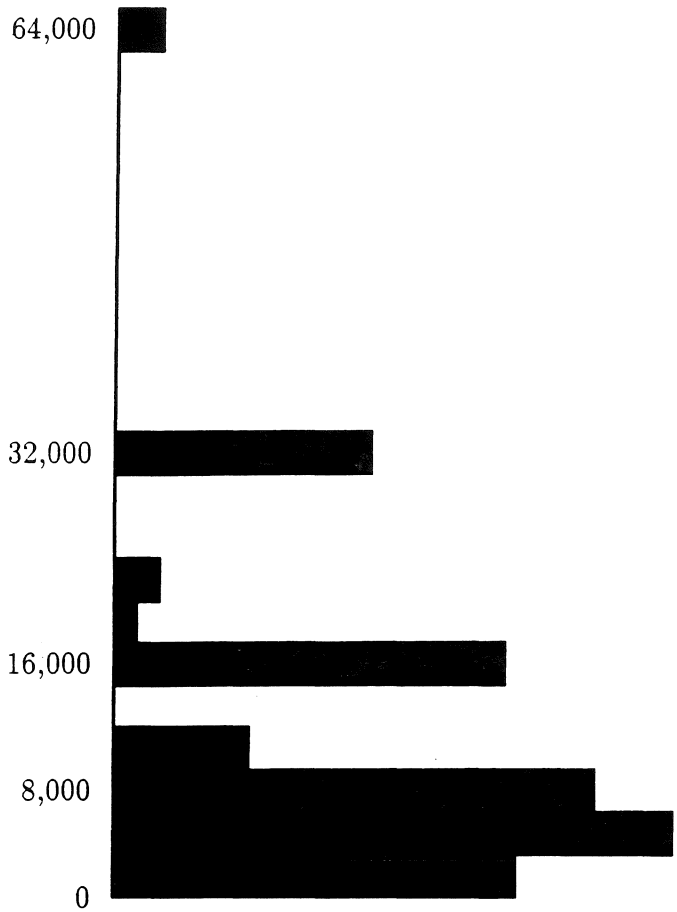


Figure 5.1: Distribution of the main memory size of 209 computers in megabytes. The four computers with 64M main memory sizes are outliers in this distribution.

difference (i.e., never greater than 0.5). Other attributes will dominate the similarity computation because they will usually have larger pairwise normalized attribute value differences. However, if the four outlier values were removed from the distribution, then the normalized difference of a randomly chosen pair of this attribute's values would increase because the values would be more evenly distributed. In summary, attributes with outliers tend to consistently contribute low values to similarity calculations, thus reducing their influence in target value predictions. Although IBL algorithms performed relatively well when using the linear normalization function, it is possible that alternatives will display better learning behavior.

The second alternative studied in this investigation is the *no-normalization* function, which eliminates the normalization process.[1] It addresses the first problem identified with linear normalization (i.e., that no normalization supports better learning behavior when the numeric attributes' ranges correspond with their relative predictive relevance). The main purpose for studying this "strawman" alternative, whose use equates the IB1 algorithm with the nearest neighbor function, is to determine whether a lesion of the normalization function negatively impacts the learning behavior of the IBL algorithms.

The third alternative, named the *standard normalization* function, addresses the second problem identified with linear normalization (i.e., outliers). This alternative divides unnormalized numeric attribute values by the standard deviation of their attribute's previously processed values. This increases the normalized difference between non-outliers and still yields large normalized differences between a non-outlier and an outlier. The standard normalization function resembles the common statistical method for standardizing a normal distribution, where the standardized normal variable is defined as the the difference of the actual and mean values divided by the standard distribution (Spiegel, 1988). However, most of the numeric predictor attributes in the databases and domains used to evaluate the IBL algorithms do not have normal distributions. Nonetheless, this alternative seems to be more intelligent than the first two. It should support better learning behavior than the linear normalization function when the distributions of the application's numeric predictor attributes contain outliers. Unfortunately, it is also expected to decrease learning rates when the attribute ranges correspond to attribute relevance.

Table 5.1 summarizes these three alternatives' results for the symbolic prediction tasks. The two LED Display domains and the Congressional Voting database are not included in this study since their attributes are all Boolean-valued. The three alternative normalization functions would yield identical learning behavior for these applications. However, the results with the two Waveform domains and the heart disease databases provide convincing evidence that the choice of the normalization function can influence the learning behavior of IBL algorithms.

The no-normalization alternative supported the best learning behavior (i.e., the highest classification accuracies) for both of the Waveform domain applications. This was expected since the ranges of the numeric-valued attributes in these domains correspond to their relative relevance for predicting accurate classifications. IB4 still recorded the highest accuracy in both domains. IB1's accuracy on the Waveform-41 domain had the greatest increase (6.3%). This increase, measured individually over the 25 trials, is significant ($t(24) = 24, p < 0.1$). The linear and standard normalization methods yield highly similar learning behavior for these applications.

[1]However, IB4's weight-learning algorithm depends on the difference between two attribute values to be in the range $[0, 1]$. Therefore, linear normalizations are computed by IB4 so that its weight-learning algorithm functions properly.

Table 5.1: Average percent accuracy ± standard error and % storage requirements for three alternative normalization functions.

Normalization	IB1	IB2	IB3	IB4
	Waveform-21			
None	75.7±1.0	70.0±1.1 30.7	74.6±1.0 11.1	78.3±1.1 13.1
Linear	75.5±1.1	68.4±1.1 32.3	74.2±1.2 11.1	76.9±1.0 13.4
Standard	74.7±1.2	67.5±0.8 33.0	73.5±1.1 12.0	76.0±0.9 13.0
	Waveform-40			
None	72.7±0.6	68.5±0.9 33.8	71.6±1.1 12.2	76.4±1.3 13.4
Linear	66.4±1.1	64.0±0.7 38.3	67.2±1.1 11.8	72.1±1.2 12.6
Standard	66.5±0.8	63.4±1.0 39.5	63.9±1.2 12.0	70.7±1.0 12.8
	Cleveland			
None	59.3±1.0	57.2±1.0 44.2	54.8±1.6 1.3	55.7±1.8 1.6
Linear	75.1±0.8	71.4±0.9 32.0	78.4±0.9 3.9	78.1±0.7 4.2
Standard	76.2±0.7	72.2±0.8 31.2	79.0±0.7 4.2	79.5±0.7 4.1
	Hungarian			
None	61.3±0.9	56.9±1.0 44.6	55.8±2.2 1.3	57.0±2.1 1.7
Linear	56.1±2.2	53.1±2.4 36.9	79.4±0.9 4.3	79.8±0.7 3.9
Standard	76.5±0.7	69.1±1.0 31.9	80.9±0.7 4.8	78.7±0.9 4.4

However, the situation is reversed in the application to the Cleveland Clinic Foundation's database. In this case, the no-normalization method lead to comparatively poor results for all the algorithms. For example, IB3's learning rate was significantly faster ($t(8) = 1.97, p < 0.05$) when it used the linear normalization method than when it did not normalize this application's instances. Its classification accuracy after processing the entire training set was also significantly higher ($t(24) = 2.64, p < 0.01$). An analysis of the distributions of the attributes in this database reveals why the no-normalization alternative performs poorly. Figure 5.2 shows the relative standard deviations of the attributes in the Cleveland database. This bar plot shows that the standard deviations of these fourteen attributes differ considerably. The standard deviation for the level of serum cholesterol, measured in mg/dl, is 51.78. The smallest standard deviations belong to the binary-valued attributes. These values are indicative of how large each attribute's pairwise differences are in the similarity computations. Since large ranges of differences affect the similarity function more than small ranges, the attributes with the larger standard deviations will have the most influence on the similarity function. The results depend on the unit used to measure each attribute's values. If the IBL algorithms were directed to measure serum cholesterol level in grams rather than milligrams per decaliter, then this attribute would have far less predictive influence. It is not obvious that the attributes with the largest standard deviations are the most relevant for predicting the presence of heart disease. Furthermore, the discrepancy in value size is so great that many of the relevant attributes are essentially ignored when the values are not normalized.

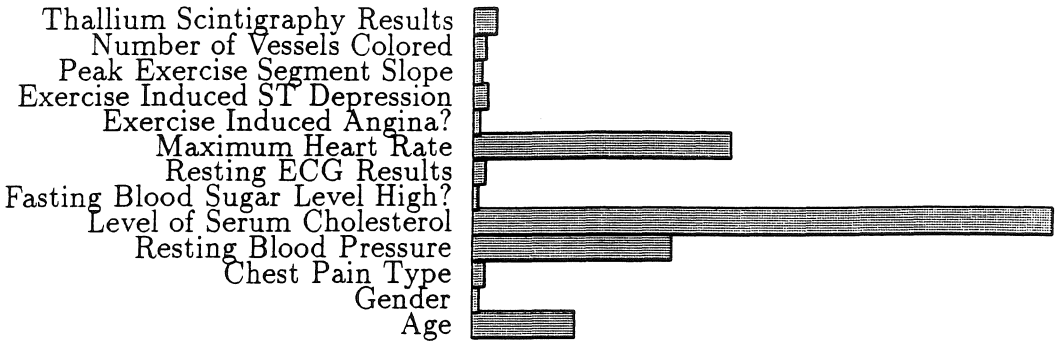


Figure 5.2: Relative standard deviations of the Cleveland database's attributes.

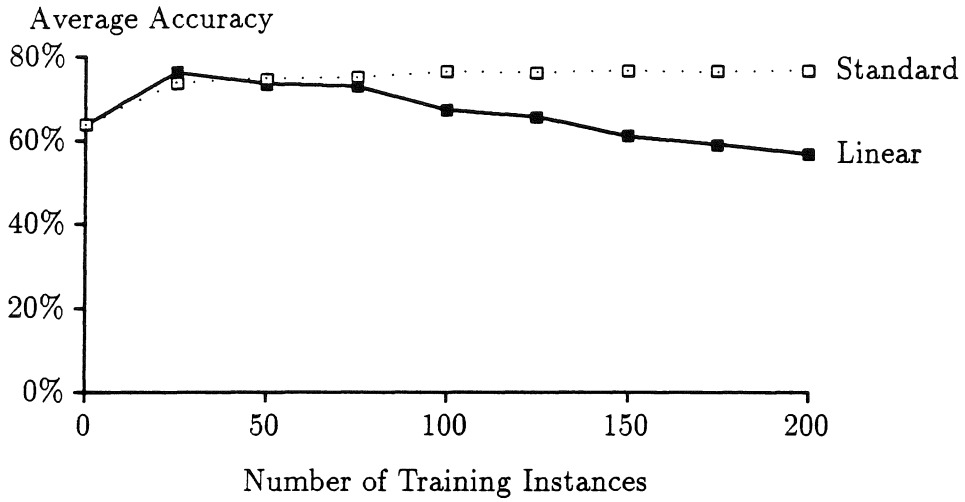


Figure 5.3: IB1's average learning curves for the Hungarian database using two different normalization algorithms.

The results with the Hungarian database are surprisingly different considering that the instances in these databases are described by the same set of attributes and the predictor attributes have similar relative standard deviations. Although the no-normalization method performed predictably poorly, as evidenced in comparisons with IB3 and IB4's performance for the other alternatives, the standard normalization alternative resulted with tremendously improved performance for the IB1 and IB2 algorithms. For example, IB1's average accuracy improved by 20%. Both its learning rate after 25 instances ($t(7) = 1.71, p < 0.1$) and its accuracy after processing the entire training set ($t(24) = 1.84, p < 0.05$) increased significantly. Figure 5.3 displays IB1's learning curves for both the linear and standard normalization methods. These results beg the question: Why does the standard normalization algorithm significantly improve the results for the Hungarian database application but

not for the Cleveland database? The main difference between these two databases is their number of missing attribute values. While the Hungarian database has 781 missing values, the Cleveland database has only six. More research is required to determine why the linear normalization method is sensitive to large numbers of missing values. It is possible that the algorithm used for processing missing attribute values, which assumes that a missing value is maximally different than the one present, affects the linear normalization method. In any case, the standard normalization method appears to be less sensitive to missing attribute values.

The three normalization strategies were also evaluated on the six numeric prediction tasks.[2] The results of these applications are summarized in Table 5.2. The no-normalization strategy's error rate was comparatively high in all but the first application, indicating that the relationship of attribute value range with its predictive relevance is small for most of these applications. This strategy's storage requirements were consistently higher for IB2 and lower for IB3 and IB4. This indicates that a flood of prediction errors occurred, which sharply increase IB2's storage requirements (i.e., causes more misclassifications) and sharply reduce IB3 and IB4's number of accepted instances (i.e., causes fewer instances to have significantly high accuracies). These applications seem to be extremely noisy from the no-normalization method's perspective. The linear normalization strategy was clearly the best in the majority of these experiments. It also recorded lower storage requirements than the standard normalization method in almost all of these applications.

The standard normalization method performed worse than expected in these applications. It is possible that the distributions of the predictor attributes in these databases are not highly skewed. If so, then the standard normalization method should not function better than the linear normalization method. Therefore, I also applied the three strategies to the relative cpu performance database to determine whether the standard normalization method would perform best in an application when all the predictor attributes had distributions such as the one shown in Figure 5.1 on page 118. Surprisingly, the linear normalization alternative again clearly recorded the lowest error rates (the no-normalization method clearly recorded the highest). This contradicted my expectations. One explanation for this behavior is that these predictor attributes' distributions with outliers correctly convey the information that most of the values (i.e., everything but the outliers) are highly similar to each other. While the linear normalization method preserves this information, the standard normalization method sacrifices it to some degree. However, when the outliers are due to noisy observations, the standard normalization method should support improved learning behavior. In summary, it is not obvious that the linear normalization method is as sensitive to the predictor attributes' distributions as suggested earlier because the outliers may not be noise. An improvement to this normalization method should increase the perceived differences of

[2]The target attribute's values were linearly normalized in these applications so that relative error rates could be generated.

Table 5.2: Average relative error ± standard error results and percent storage requirements for the six numeric prediction tasks. The three lines for each application refer to using the linear, no-normalization, and standard normalization algorithms respectively.

Strategy	IB1	IB2		IB3		IB4	
	Hungarian						
Linear	0.39±0.022	0.43±0.026	36.7	0.23±0.013	6.3	0.20±0.008	7.5
None	0.45±0.023	0.44±0.026	45.2	0.37±0.016	5.1	0.35±0.020	6.0
Standard	0.24±0.008	0.30±0.010	30.8	0.21±0.009	7.9	0.22±0.009	7.6
	Cleveland						
Linear	0.19±0.005	0.20±0.005	52.9	0.19±0.005	14.2	0.18±0.005	13.9
None	0.43±0.051	0.41±0.044	75.8	0.31±0.030	5.7	0.26±0.020	6.5
Standard	0.21±0.006	0.22±0.005	53.3	0.19±0.006	15.8	0.20±0.004	15.4
	Horsepower						
Linear	0.04±0.002	0.06±0.003	24.9	0.08±0.004	16.8	0.07±0.003	17.8
None	0.18±0.017	0.19±0.019	89.6	0.28±0.022	6.3	0.27±0.020	5.7
Standard	0.09±0.004	0.10±0.004	34.2	0.11±0.006	19.7	0.12±0.007	19.4
	Number of Months to Live						
Linear	0.20±0.006	0.22±0.006	57.4	0.20±0.005	18.4	0.20±0.007	17.3
None	0.35±0.013	0.36±0.016	74.3	0.31±0.012	9.7	0.30±0.015	11.2
Standard	0.25±0.008	0.26±0.008	64.4	0.24±0.008	19.1	0.23±0.008	19.5
	Serum Cholesterol Level						
Linear	0.14±0.007	0.15±0.007	52.3	0.13±0.007	13.6	0.14±0.007	13.3
None	0.15±0.017	0.16±0.016	63.6	0.17±0.013	4.3	0.17±0.014	5.0
Standard	0.20±0.007	0.20±0.007	60.4	0.18±0.007	19.1	0.19±0.006	17.8
	Tumor Size						
Linear	0.22±0.004	0.23±0.004	62.0	0.21±0.003	16.1	0.21±0.003	16.5
None	0.22±0.012	0.21±0.010	73.1	0.21±0.006	5.7	0.20±0.006	6.4
Standard	0.23±0.005	0.24±0.004	65.4	0.22±0.004	18.4	0.20±0.004	18.9

the majority of attribute values when the instances' containing the outlier values appear to be noisy.

This study showed that the choice of the normalization method used in IBL algorithms *can* significantly affect their learning behavior. The no-normalization method worked well when the predictor attributes' ranges matched their relative relevance for predicting target values. However, both the linear and standard normalization methods performed significantly better than no normalization in the clear majority of applications. The linear normalization method performed best in the numeric prediction applications. Although more

research is required to characterize the weaknesses of (and subsequently improve) the linear normalization method, it is a better alternative than no normalization without prior information concerning attribute relevance.

5.2 Alternative Performance Components

This section describes high-level parameter studies with the functions composing the IBL framework's performance component. The purpose of these studies is to investigate some alternative and intuitively pleasing component definitions to determine whether they lead to improved learning behavior. Section 5.2.1 describes two studies with alternative similarity functions. Both investigations are similar to the one in Section 5.1 in that they compare three methods for accomplishing the same task. The first study concerns the processing of missing attribute values while the second concerns the processing of symbolic attribute values. Section 5.2.2 describes a study of the IBL algorithms' behavior when using a different prediction function, one where more than one instance is used to derive target value predictions (i.e., $k > 1$).

5.2.1 Alternative Similarity Functions

Missing Attribute Values

Why would any data collector want to include missing attribute values in their database? Typical purposes for collecting databases such as the ones in the UCI repository are to discover regularities in the data and to derive accurate predictions for future data points. The presence of missing attribute values rarely assists in this process. Nonetheless, many databases contain missing attribute values. For example, 27 of the 49 databases in the UCI repository contain missing attribute values.[3] Therefore, inductive learning algorithms must be able to process instances with missing values if they are going to be broadly applicable. It's possible that pre-processing methods can be used to eliminate missing attribute values, but this also constitutes a means for processing them and is, therefore, a choice point in the design of the learning algorithm.

There are many ways to process missing attribute values in inductive learning algorithms. For example, Gams and Lavrač (1987) noted that Assistant-86 (Cestnik, *et al.*, 1987), C4 (Quinlan, 1987), CN2 (Clark & Niblett, 1989), AQ15 (Michalski *et al.*, 1986), and CART

[3]This does not include dataset generators, toy domains, and domain theories.

(Breiman *et al.*, 1984) all process missing attribute values differently. Methods include ignoring them, replacing them by most probable values, processing them as a separate symbolic value, replacing them by all possible values, and replacing them with value-probability pairs.

Missing attribute values pose a problem for the similarity function in algorithms based on the nearest neighbor pattern classifier. Unfortunately, I have not (yet) found any published study, formal or informal, on methods for processing missing attribute values in nearest neighbor algorithms or similarity functions. Several potential methods seem intuitively sensible. For example, the default method used in the four IBL algorithms, which I will refer to as the *MaxDiff* algorithm, assumes that missing attribute values are as different as possible from known attribute values in pairwise attribute-value comparisons. This prevents dissimilar instances from being used inappropriately as a target value predictor for novel instances. It worked relatively well for the IBL algorithms, which recorded predictive accuracies similar to other supervised learning algorithms (Kibler & Aha, 1987; 1988; Aha & Kibler, 1989). However, the MaxDiff algorithm was never evaluated against alternatives to determine whether it supports comparatively good learning behavior.

One obvious alternative is to replace the missing value with the most probable attribute value among the previously observed values, as was done in CN2. Similarly, missing numeric attribute values are replaced with the average of their observed values. This method, named *Mode-Mean*, is the second alternative investigated in the following empirical study. This alternative acts as a foil to the maximum difference method; similarities computed using Mode-Mean can be *greater* than the two instance's actual similarity, which can be computed only when all attribute values are known.

The third alternative, which I will refer to as the *Ignore* method, amounts to a lesion study. It operates by essentially ignoring the missing attribute value. However, ignoring an attribute cannot be distinguished from assuming that the two values are identical, which yields the minimum potential difference between the two instances and, subsequently, yields their maximum potential similarity. Therefore, the Ignore algorithm normalizes the similarity function's result; it divides the computed distance by the square root of the number of attributes that contribute to the distance measure (i.e., the number of attributes whose values are known in both instances):[4]

$$\text{Similarity}(x, y, t, P) = \frac{1}{\text{Distance}(x, y, t, P)} \tag{5.1}$$

$$\text{Distance}(x, y, t, P) = \frac{\sqrt{\sum_{i \in P} \text{Both\_known}(x_i, y_i) \times \text{Attribute\_difference}(x_i, y_i)}}{\sqrt{\sum_{i \in P} \text{Both\_known}(x_i, y_i)}} \tag{5.2}$$

[4]An additional value of 0.001 is added to the denominator in Equation 5.2 to prevent division by zero.

Table 5.3: Average percent accuracy ± standard error and percent storage requirements for two symbolic prediction tasks. Three alternatives for processing missing attribute values are compared.

Method	IB1	IB2	IB3	IB4
	Hungarian database			
MaxDiff	56.1±2.2	53.1±2.4 36.9	79.4±0.9 4.3	79.8±0.7 3.9
Mode-Mean	76.1±0.8	69.2±1.0 29.3	80.0±0.7 4.3	79.7±0.6 3.6
Ignore	74.8±0.7	70.1±0.8 29.9	81.3±0.7 4.5	79.6±0.7 4.1
	Congressional Voting database			
MaxDiff	91.8±0.4	90.9±0.5 11.6	90.6±0.6 3.5	93.8±0.4 3.0
Mode-Mean	93.4±0.4	88.9±0.6 12.6	91.5±0.8 3.5	94.3±0.5 3.5
Ignore	88.6±0.7	86.3±0.7 12.8	89.9±0.8 3.6	91.8±0.9 3.5

where

$$\text{Both\_known}(x_i, y_i) = \begin{cases} 1 & \text{if the values of both } x_i \text{ and } y_i \text{ are known} \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

Only two of the databases and used for the seven symbolic prediction tasks contain substantial numbers of missing attribute value (i.e., Hungarian and Congressional voting databases are missing 781 and 288 values respectively). Table 5.3 summarizes the results of applying the three alternatives in these two experiments.

The most noticeable result is that the Mode-Mean ($t(24) = 1.88, p < 0.05$) and Ignore ($t(24) = 1.97, p < 0.05$) strategies significantly increased IB1's classification accuracy in the Hungarian heart disease application. Naturally, IB2's accuracy also increased by the same amount. The Mode-Mean and Ignore strategies performed about equally well in this application.

However, the Mode-Mean strategy always recorded higher classification accuracies than the Ignore strategy for the Congressional Voting database. This increase was significant for IB1 ($t(24) = 1.67, p < 0.1$). Based on these two applications, replacing by average or most frequent value appears to the most resilient of the three strategies for tolerating missing attribute values.

Unfortunately, this is not true in general. The Mode-Mean strategy performs poorly in several variants of the LED-7 Display domain application examined previously by Quinlan (1989). Quinlan's intention was to investigate how variants of C4 performed when varying amounts of missing values occurred for each attribute. The first variant he investigated is the basic LED-7 Display domain application except that attribute values were deleted with a probability of 25%. Next, he investigated the *step* variant, which applies an unknown

Table 5.4: IB1's average percent accuracy ± standard error for three variants the LED-7 Display domain task. Quinlan (1989) applied seven variants of C4 to these applications. The results for the best and worst-performing variation of C4 for each application are also shown in this table.

Algorithm	25%	Step	Slope
IB1 MaxDiff	51.0±1.3	46.8±1.2	41.2±1.3
IB1 Mode-Mean	39.0±1.1	39.2±1.0	31.3±1.3
IB1 Ignore	54.8±1.2	44.5±1.2	43.3±1.4
C4 best	56.2±0.3	50.6±0.6	38.0±0.6
C4 worst	49.2±0.9	45.0±1.1	33.4±1.2

rate of 50% to four of the attributes[5] and 0% to the other three. Finally, Quinlan's *slope* variant applies a different probability to each attribute. The missing probability rate begins at 10% for the first attribute and increases by 10% to a maximum of 70% for the seventh (and final) predictor attribute. Table 5.4 summarizes these three studies. It includes the results of IB1, averaged over 25 trials for each of the three methods for tolerating missing values, and the best-performing variant that Quinlan found in each of the studies among the seven variants he evaluated. The Mode-Mean strategy for tolerating missing values performed worse than the Ignore strategy in all three applications, significantly worse for the 25% missing ($t(24) = 2.7, p < 0.01$) and slope ($t(24) = 2.3, p < 0.025$) applications. The Mode-Mean strategy was also the only strategy to fall below the range of results reported for C4 in all three applications. Interestingly, the other two strategies were far superior than C4's best for the slope application, suggesting that IBL algorithms can tolerate varying amounts of missing attribute values better than can decision tree algorithms. However, C4's best is clearly better than IB1's best for the other two applications.

The noise in these applications probably significantly affects the behavior of these three alternative algorithms for processing missing attribute values. I investigated the possibility that the Mode-Mean strategy's poor performance was due to the noise rather than to the missing attribute values. Figure 5.4 shows the average of the classification accuracies attained by IB1 when using the three missing value strategies on the LED-7 Display domain when the noise level was 0%. These results confirm that Mode-Mean does not perform well on this domain even when there is no attribute noise.

The reason that the Mode-Mean strategy performs poorly on the LED-7 Display domain variants is because it frequently overestimates two instance's similarity. The maximum difference strategy assumes that two instances are maximally different, thus never overestimating their actual similarity. The Ignore strategy essentially ignores attributes with missing values and normalizes by the number of attributes whose values are known in both instances.

[5]Quinlan (1989) didn't say which set of four attributes, or whether it varied. I chose the first four in my experiments.

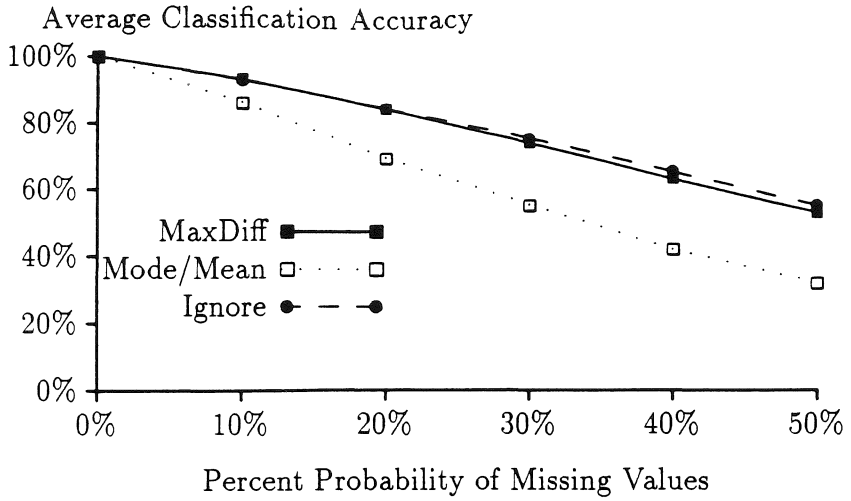


Figure 5.4: IB1's average classification accuracies on the LED-7 Display domain. This graph compares three strategies for processing missing attribute values.

Although this strategy can yield higher values than the actual similarity, Mode-Mean will yield even higher values when the replacement value is highly similar to the missing value. Furthermore, when using the Mode-Mean strategy in a domain with a sufficient number of missing attribute values, the law of large numbers assures that at least one of the training instances replacement values will make a less similar instance look highly similar to the target instance (i.e., the instance being classified). In such cases, a clearly dissimilar instance is responsible for the classification, which can often cause prediction errors. This cannot occur as easily with the other strategies because the worst that they can do is make the most similar instance look less similar, thus replacing the best classifying instance with one that is, in general, only slightly less similar than it to generate the target value prediction. The Mode-Mean strategy should be especially susceptible to Boolean-valued attributes, where it can often replace the missing value by a value that matches the other instance's value. Mode-Mean should (and did) perform better in applications with numeric- rather than with Boolean-valued attributes. Finally, Mode-Mean assumes that a pair of missing values have the same value. This default assumption can occur often in applications and can easily modify the set of instance(s) used to generate target value predictions.

Table 5.5 describes how the three strategies fared in the subset of numeric prediction tasks that whose target function is taken from a database containing substantial numbers of missing values (i.e., 59, 288, and 106 respectively). Since the majority of predictor attributes in these three applications are numeric-valued, it's not surprising that the Mode-Mean strategy performed comparably well. As expected, both the Mode-Mean and Ignore strategies repaired IB1 and IB2 in the Hungarian heard disease diagnosis application. Otherwise, the three algorithms appear to be equally good choices for processing missing attribute values.

Table 5.5: Average percent accuracy ± standard error and percent storage requirements for the three numeric prediction tasks with substantial numbers of missing attribute values. Three alternatives for processing missing attribute values are compared.

Method	IB1	IB2	IB3	IB4
	Horsepower			
MaxDiff	0.04±0.002	0.06±0.003 24.9	0.08±0.004 16.8	0.07±0.003 17.8
Mode-Mean	0.04±0.002	0.06±0.003 24.8	0.07±0.003 17.6	0.08±0.003 18.0
Ignore	0.04±0.002	0.06±0.003 25.1	0.07±0.003 17.4	0.08±0.003 18.6
	Hungarian Heart Disease			
MaxDiff	0.39±0.022	0.43±0.026 37.0	0.23±0.013 6.3	0.20±0.008 7.5
Mode-Mean	0.23±0.008	0.30±0.011 28.2	0.21±0.010 7.5	0.20±0.008 7.7
Ignore	0.24±0.008	0.29±0.012 27.9	0.21±0.009 7.4	0.21±0.008 7.6
	Number of Months to Live			
MaxDiff	0.20±0.006	0.22±0.006 57.4	0.20±0.005 18.4	0.20±0.007 17.3
Mode-Mean	0.19±0.006	0.21±0.006 55.7	0.20±0.006 19.9	0.20±0.006 18.5
Ignore	0.20±0.006	0.22±0.005 58.3	0.20±0.006 19.1	0.20±0.006 18.4

In summary, none of these three strategies consistently supported superior learning behavior. The default MaxDiff algorithm performed poorly in the Hungarian database application. The Ignore strategy performed poorly in the application to the Congressional Voting database. Finally, the Mode-Mean strategy performed poorly when applied to variants of the LED-7 Display domain. The choice for how an IBL algorithm tolerates missing attribute values can lead to significantly improved or significantly worse learning behavior. However, further research will be needed to carefully characterize the conditions under which these alternatives will perform well.

Symbolic Attribute Values

Almost none of the predictor attributes in the two sets of prediction tasks are symbolic-valued. Also, the mathematical analysis in Chapter 3 concerns only numeric-valued attributes. Nonetheless, prediction systems based on IBL algorithms have performed well in a variety of learning and prediction tasks whose instances are described by symbolic attribute values.

This section compares three algorithms that define similarity for symbolic-valued attributes. These algorithms vary in their definitions of Symbolic_difference, which is a subfunction of the Attribute_difference function. This function is defined for IB1 as follows:

$$\text{Similarity}(x, y, t, P) = \frac{1}{\sqrt{\sum_{i \in P} \text{Attribute\_difference}(x_i, y_i)}} \tag{5.4}$$

$$\text{Attribute\_difference}(x_i, y_i) = \begin{cases} (x_i - y_i)^2 & i \text{ is numeric-valued} \\ \text{Symbolic\_difference}(x_i, y_i) & \text{otherwise} \end{cases} \tag{5.5}$$

The three strategies to be examined were originally introduced by Stanfill and Waltz (1986; Stanfill, 1987), who described an application of their MBRtalk system to a phoneme and word pronunciation task. The first function, which they named the *overlap metric*, is the default method used in IB1 through IB4. It is the identity check, defined as

$$\text{Symbolic\_difference}(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i \\ 1 & \text{otherwise} \end{cases} \tag{5.6}$$

This simple function, like the two alternatives described in the following paragraphs, supports several valuable relational properties. These include a unit range, symmetry, identity, and transitivity:

$$\text{Similarity}(a, a) = 0 \tag{5.7}$$

$$1 \geq \text{Similarity}(a, b) \geq 0 \tag{5.8}$$

$$\text{Similarity}(a, b) = \text{Similarity}(b, a) \tag{5.9}$$

$$\text{Similarity}(a, b) + \text{Similarity}(b, c) \geq \text{Similarity}(a, c) \tag{5.10}$$

The overlap metric is an oversimplified definition. One extension described by Stanfill and Waltz is the *weighted-features metric*. It also defines $\text{Similarity}(a, a) = 0$, but doesn't define $\text{Similarity}(a, b) = 1$ when $a \neq b$. Instead, it assigns weights to each attribute based on their observed ability to constrain the value of the target attribute among previously processed instances.[6] This is a necessary property for a similarity function in their phoneme pronunciation task, where an instance had seven predictors (i.e., the three preceding letters, the letter of concern, and the three following letters in the word) and two target ("goal") fields (i.e., phoneme pronunciation and stress). Without this capability of differential weighting, each of the predictors would count equally towards the pronunciation of the for the letter of concern. For example, the overlap metric yields equal similarity scores to instances in the database that matched in all letters but the one in focus with instances that matched

[6]Medin and Schaffer (1978) used this same pairwise function in their exemplar-based context model, although their similarity function was a product rather than a sum of the pairwise similarity values. Like Stanfill and Waltz, their intention was to capture the interrelationships of the predictor attributes.

in all but the third following letter. Obviously, matching the letter whose phoneme is to be predicted is often more valuable than matching the third following letter. The weighted features metric is described in Equation 5.11, where I've assumed that the three additional parameters are passed by Attribute_difference. Let t be the target attribute, Domain(t) be the set of values that may be assigned to t, y be a stored instance in t's partial concept description, and $T(x)$ be the set of training instances processed before processing the current instance x. Then

$$\text{Symbolic\_difference}(x, y, i, t, T(x)) = \begin{cases} 0 & \text{if } x_i = y_i \\ \text{Weight}(x, i, t, T(x)) & \text{otherwise} \end{cases} \tag{5.11}$$

where the weight function is defined as

$$\text{Weight}(x, i, t, T(x)) = \sqrt{\sum_{v \in \text{Domain}(t)} \left(\frac{|\{z \in T(x) | z_i = x_i \wedge z_t = v\}|}{|\{z \in T(x) | z_i = x_i\}|} \right)^2} \tag{5.12}$$

This metric fulfills some of the same purpose fulfilled by IB4: it weights attributes differentially according to their estimated relevance for generating accurate predictions. Let $T(x|x_i)$ be the set of processed training instances whose value for predictor attribute i is x_i. This metric resembles Quinlan's (1986a) information gain function as used in ID3 and C4: It is maximized when the instances in $T(x|x_i)$ all have the same value for target attribute t and minimized when the the target attribute's values in $T(x|x_i)$ are evenly distributed. Thus, this metric measures the constraint that value x_i places on the possible values of x_t.

Although the weighted features metric is more intuitively pleasing than the overlap metric, it fails to exploit partial ordering information inherent in an attribute's range of symbolic values. For example, if an attribute's set of symbolic values is {blue, green, tiger}, then we can imagine an ordering such that the similarity of the first two values is higher than the similarity of either to the third. In extreme cases, two of the attribute's symbolic values may be synonymous while another may be their antonym. This highlights the problem with the weighted features metric: its output should correspond to the *degree* to which the given pair of symbolic values differ.

Therefore, Stanfill and Waltz proposed an extension of the weighted features metric named the *value difference metric.* It defines a similarity matrix for a predictor's range of values by comparing each pair of values' corresponding distribution of target values. That is, given symbolic values x_i and y_i, it compares the distribution of the target values in $T(x_i)$ with the distribution of target values in $T(y_i)$. This metric still retains the notion of an attribute weight, which is multiplied by this measure of the values' difference. Equation 5.14 summarizes the value difference metric, where the Weight function was defined in Equation 5.12.

$$\text{Symbolic\_difference}(x, y, i, t, T(x)) = \tag{5.13}$$

$$\begin{cases} 0 & \text{if } x_i = y_i \\ \text{Weight}(x, i, t, T(x)) \times \text{Difference}(x, y, i, t, T(x)) & \text{otherwise} \end{cases} \tag{5.14}$$

where the difference function is defined as

$$\text{Difference}(x, y, i, t, T(x)) = \tag{5.15}$$

$$\sum_{v \in \text{Domain}(t)} \left(\frac{|\{z \in T(x|x_i)|z_t = v\}|}{|\{z \in T(x|x_i)\}|} - \frac{|\{z \in T(x|y_i)|z_t = v\}|}{|\{z \in T(x|y_i)\}|} \right)^2 \tag{5.16}$$

Stanfill and Waltz (1986) used the value difference metric in their MBRtalk system, which predicts phoneme pronunciations and stress. In one of their experiments, its predictions were evaluated by a set of human judges against a dictionary's predictions on a set of 100 test words after MBRtalk was trained on a 4438-word subset of the *Merriam-Webster's Pocket Dictionary*. MBRtalk was judged to have correctly pronounced 47% of the test words, which is quite an accomplishment considering that words consist of several phonemes. Most of the errors were due to lack of domain knowledge (e.g., grammatical information, word usage).

The authors of MBRtalk didn't compare their algorithm against other learning algorithms. However, Cost and Salzberg (1990) recently compared PEBLS, an IBL algorithm that uses a variant of the value difference metric, to several other learning algorithms on three well-known databases. They found that their algorithm recorded higher classification accuracies than reported by Qian and Sejnowski (1988) for predicting protein structure, performed as well as KBANN (Towell, Shavlik, & Noordewier, 1990) on their promoter sequence database, and better than the back propagation, ID3, and perceptron training algorithms on the NETtalk database (Sejnowski & Rosenberg, 1987) when using a local output encoding. Although their prediction function differs from the one used in MBRtalk (e.g., PEBLS uses a 1-nearest neighbor while MBRtalk uses a k-nearest neighbor prediction function), these results suggest that the value difference metric supports excellent learning behavior in IBL algorithms.

Nonetheless, neither Stanfill and Waltz nor Cost and Salzberg compared the value difference metric with the other two metrics. Is it possible that the value difference metric doesn't always support the best learning behavior among these three algorithms? I investigated this possibility by comparing them a set of symbolic prediction tasks. This set of tasks does not include the six tasks used previously to study the IBL algorithms behavior because they contained few symbolic predictor attributes. Instead, the applications included:

1. The automobile database from which the horsepower target function was taken. The target function is the car's assigned insurance risk rating, which has seven possible values. Five (20%) of this database's twenty-five predictor attributes are symbolic-valued.
2. The breast cancer database donated by the Ljubljana Institute of Oncology. The target function is whether the patient experienced signs of cancer recurrence. Three (33%) of this applications' nine predictor attributes are symbolic-valued.

Table 5.6: Average percent accuracy ± standard error and percent storage requirements for three symbolic prediction tasks. Three alternatives for processing symbolic attribute values are compared.

Metric	IB1	IB2	IB3	IB4
	Auto Imports Database			
Overlap	70.8±1.0	65.5±0.9 46.3	39.1±2.3 7.9	24.3±2.7 7.1
Weighted	66.4±1.0	61.3±1.0 50.0	37.4±2.0 7.8	22.3±2.2 6.9
ValueDiff	67.7±1.1	57.4±0.8 44.0	47.7±1.8 12.0	25.2±2.7 9.0
	Breast Cancer Database			
Overlap	65.4±1.0	60.8±0.9 38.9	60.0±3.2 1.6	60.2±3.1 2.0
Weighted	66.2±1.0	62.5±1.1 39.9	54.7±3.6 1.8	59.8±3.1 2.2
ValueDiff	67.0±1.0	62.8±1.0 38.3	57.7±3.9 2.0	54.0±3.4 1.9
	Lymphography Database			
Overlap	79.6±1.0	74.4±1.5 31.2	37.7±4.4 1.9	38.8±4.0 2.9
Weighted	80.5±1.3	73.4±1.3 31.0	44.5±4.6 2.0	38.5±3.3 3.5
ValueDiff	76.9±1.3	71.0±1.0 27.7	43.8±5.4 2.4	46.7±3.2 4.8

3. The lymphography database, which was also donated by the Ljubljana Institute of Oncology. The target function is patient diagnosis (four possible values). Six (33%) of the application's 18 predictor attributes are symbolic-valued.

The results of these experiments are summarized in Table 5.6. Published results exist for the latter two applications (e.g., Clark & Niblett, 1989). Typical classification accuracies for predicting breast cancer recurrence range from 65% (a simple Bayes algorithm) to 72% (a simplification of Michalski's (*et al.*, 1986) AQ15 algorithm). Accuracies for the lymphography database range from 76% (AQ11) to 83% (Bayes). IB1's performance is in both of these ranges.

Unfortunately, few conclusions can be drawn from analyzing IB3 and IB4's learning behavior in these domains. These algorithms did not accept sufficient numbers of instances to allow them to record good classification accuracies. This was expected for the automobile database since this occurred when IB3 and IB4 tried to predict horsepower. These IBL algorithms' stringent requirements for instance acceptance must be relaxed before they will perform well in these applications.

Fortunately, the results with IB1 and IB2 are more useful for comparing the three metrics' capabilities in these applications. Somewhat surprisingly, the value difference and weighted features metric did not outperform the simple overlap metric in these applications. In fact, the overlap metric scored the highest average classification accuracy for the automobile database. It also scored the highest accuracy for IB2 in two of the three applications. These results contradict the intuitions that the weighted features and value difference metrics will outperform the overlap metric in most applications.

Table 5.7: Average relative error and percent storage requirements for two numeric prediction tasks. Three alternatives for processing symbolic attribute values are compared.

Metric	IB1	IB2	IB3	IB4
	Tumor Size Target Function			
Overlap	0.22±0.004	0.23±0.004 62.0	0.21±0.003 16.1	0.21±0.004 16.6
Weighted	0.22±0.003	0.23±0.003 62.7	0.21±0.004 17.1	0.21±0.004 15.7
ValueDiff	0.21±0.003	0.22±0.003 61.7	0.21±0.004 16.9	0.21±0.003 16.0
	Horsepower Target Function			
Overlap	0.04±0.002	0.06±0.003 24.9	0.08±0.004 16.8	0.07±0.004 16.0
Weighted	0.05±0.003	0.06±0.003 25.7	0.08±0.004 16.4	0.06±0.003 16.5
ValueDiff	0.05±0.003	0.07±0.003 25.1	0.07±0.003 16.8	0.07±0.003 16.3

An analysis of the database's symbolic predictor attributes shows that some of them clearly have values that are more similar to each other than to others (e.g., the automobile's *make* attribute includes both high-priced and economy manufacturers). Therefore, there was ample opportunity for the weighted features and value difference metrics to outperform the overlap metric. It is probable that the large number of numeric predictor attributes in these databases may have reduced the impact of the three similarity functions studied in this experiment. Nonetheless, the point of this experiment is that the overlap metric is not always a bad choice for processing symbolic attribute values.

I also applied these algorithms to the two (of the six) numeric prediction tasks that contain substantial numbers of symbolic predictor attributes. These include the tumor-size and horsepower prediction tasks, whose symbolic attributes account for 44.4% and 24% of the predictors respectively. The results, summarized in Table 5.7, also suggest that the overlap metric is as useful as the others in many applications. However, this isn't surprising considering that these target functions were taken from two of the same datasets used in the symbolic prediction tasks (i.e., Breast Cancer and Auto Imports databases). Nonetheless, these results confirm the need for further analyses of metrics for processing symbolic attribute values in instance-based learning algorithms. The next step is to evaluate these three metrics on the larger applications investigated by Stanfill and Waltz (1986) and Cost and Salzberg (1990). This will remain as future research.

5.2.2 Alternative Prediction Functions

All of the previous studies have used variants of the nearest neighbor prediction function. This function is given a set of similarity measures and uses a single stored instance as the basis for which to generate target value predictions. This section examines the k-nearest neighbor prediction function, where $k > 1$.

The k-nearest neighbor (k-NN) algorithm has been both theoretically analyzed and empirically tested in the pattern classification and machine learning literatures. Duda and Hart (1973) showed that, as k increases to infinity, the k-NN algorithm's accuracy increases until it reaches the Bayes optimal limit. This suggests that IBL algorithms based on the k-NN prediction function may also generate more accurate concept descriptions. However, the domain examined in Duda and Hart's analysis was noise-free, was described only by relevant attributes, and was infinite in size. Can algorithms based on the k-nearest neighbor prediction function perform well in practical prediction tasks?

The answer in the pattern recognition literature is mixed. Tomek (1976b) investigated Wilson's (1972) k-NN-editing algorithm. This algorithm is a foil of IB2; the only instances saved are those whose target values are *correctly* predicted. Tomek applied it to an artificial domain and described empirical evidence that the k-NN editing algorithms improve predictive accuracy. However, Gates (1972) investigations convinced him that k-NN editing algorithms offered "no further improvements" in performance. He investigated k-NN versions of Hart's (1968) CNN algorithm, which is much more closely related to IB2 since it stores only the instances whose target values are *incorrectly* predicted. Gates applied variants of the CNN algorithm to R. A. Fisher's (1936) iris database. He found that k-NN editing sharply increases storage requirements without consistently improving predictive accuracy. This result isn't surprising considering the nature of the iris database. One of its three classes is linearly separable from the others, which are nearly separable from each other. It is difficult to improve on the high classification accuracies obtainable using the CNN algorithm. Can the k-NN prediction function improve the learning behavior of IBL algorithms in other applications?

The answer is a resounding *yes*. For example, the ALFA system (Jabbour, Riveros, Landsbergen, & Meyer, 1987) uses an 8-nearest neighbor function to predict the power load required in three central New York State cities for the Niagra Mohawk Power Company. Its average accuracy is as good as that of human experts but requires two orders of magnitude less time to generate. Stanfill and Waltz (1986) used a 10-nearest neighbor algorithm in their MBRtalk algorithm, which performed well in a word pronunciation and stress prediction task. Connell and Utgoff (1987) used a weighted k-NN prediction function in their CART algorithm in an application to the cart-and-pole task. While other methods for solving this task have required between 75 and thousands of learning trials, CART learned to balance a pole after an average of less than than 16 learning trials. Finally, Moore (1990) used a k-NN algorithm to combat noise in his application to robotic manipulation tasks. There is no doubt that a k-NN prediction function can support good learning behavior in applications to a large variety of tasks.

However, these applications were case studies. Few comparisons of the nearest neighbor and k-NN algorithms have been published (e.g., Aha & Kibler, 1989). This section compares these algorithms on the same two sets of applications used earlier by varying the value of k from one to eleven by two. Although this is only a subset of the possible settings for k,

Table 5.8: Average percent accuracy ± standard error and percent storage requirements for four symbolic prediction tasks for six settings of k.

Database	k	IB1	IB2	IB3	IB4
Hungarian	1	56.1±2.2	53.1±2.4 36.9	79.4±0.9 4.3	79.8±0.7 3.9
	3	61.2±2.2	57.4±2.0 32.4	49.7±3.7 0.9	51.2±3.9 1.4
	5	65.2±2.1	65.7±1.5 30.2	46.5±2.6 0.6	43.8±3.0 1.3
	7	68.7±1.6	66.1±1.8 29.0	48.8±2.9 0.8	46.0±3.0 0.7
	9	70.8±1.6	67.3±1.5 28.1	46.0±2.4 0.6	47.4±2.3 0.4
	11	71.6±1.4	69.1±1.6 28.6	49.5±2.4 0.5	46.8±2.3 0.9
Voting	1	91.8±0.4	90.9±0.5 11.6	90.6±0.6 3.5	93.8±0.4 3.0
	3	92.4±0.4	92.5±0.6 11.7	61.8±4.7 2.1	58.2±4.7 1.3
	5	92.8±0.4	92.6±0.6 11.7	46.4±2.3 0.5	47.6±2.4 0.4
	7	92.9±0.4	92.5±0.4 11.8	50.8±2.5 0.5	42.8±1.8 0.6
	9	92.6±0.4	93.0±0.6 12.2	48.1±2.1 0.7	46.7±2.4 0.5
	11	92.5±0.4	93.0±0.6 9.3	46.7±2.3 0.4	47.1±2.0 0.5
LED-25	1	47.9±0.6	43.7±0.8 60.9	46.6±0.7 25.3	66.1±0.6 25.5
	3	57.7±0.8	50.1±0.6 55.4	52.4±0.8 26.1	65.4±2.3 32.2
	5	60.5±0.9	54.4±0.9 52.6	55.2±1.0 23.8	49.8±7.6 33.3
	7	65.2±0.8	55.8±0.6 50.7	55.1±0.9 20.6	26.2±5.3 30.0
	9	66.1±0.6	57.7±0.7 49.7	42.8±4.1 14.6	21.6±4.8 26.8
	11	67.2±0.6	57.9±0.7 49.1	32.8±4.5 9.6	16.9±3.8 27.7
Waveform-40	1	68.6±0.7	64.0±0.7 38.3	67.2±1.1 11.8	72.1±1.2 12.6
	3	72.3±1.0	65.7±0.9 34.9	61.7±2.8 7.6	52.1±3.4 7.8
	5	74.7±0.9	69.0±1.0 33.7	43.2±3.5 3.3	37.8±2.2 3.8
	7	76.0±0.9	69.4±1.2 32.7	37.2±2.1 1.3	36.4±1.6 2.2
	9	75.8±0.9	70.5±1.1 31.7	34.4±1.6 1.2	33.9±1.1 2.1
	11	76.5±0.6	72.4±0.9 31.1	35.1±1.1 0.7	34.6±1.4 1.4

which can be set as high as the size of the training set, they were sufficient to show that a repeating pattern of learning behavior emerges as k is increased. Table 5.8 summarizes the results for a representative subset of the symbolic prediction tasks. Figure 5.5, which summarizes the results for the Cleveland database, lends more intuition to the behavior of the IBL algorithms as k is increased. In general, the experiments show that IB1 and IB2's accuracies increase as k increased, peaked, and decrease for higher values of k. The value of k for peak performance differs depending on the application. However, IB3 and IB4, as currently defined, are extremely sensitive to the value of k. They perform poorly when $k > 1$. The following paragraphs describe these results in more detail.

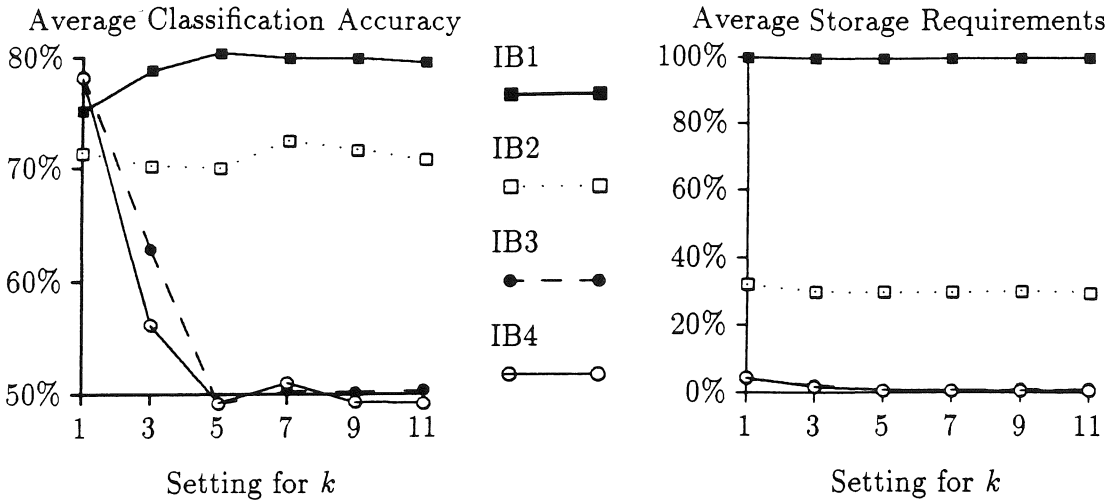


Figure 5.5: Average classification accuracy and storage requirements for the Cleveland database as k is varied.

Noisy artificial domains:

The results for the LED-7 and Waveform-21 applications were similar to the results shown in Figure 5.5 for the Cleveland database. IB1's behavior was impressive. Its accuracy increased with increasing values for k for the LED-7 application until $k = 5$, where it recorded a 74.2% average classification accuracy. This is 1.7% higher than IB3's best classification accuracy (at $k = 1$) and 5.2% higher than C4's average classification accuracy. In fact, this is even higher than the Bayes optimal classification rate of 74%, which suggests that no empirical learning algorithm will outperform IB1 in this domain. IB1's accuracy then gradually decreased to 68.9% when $k = 11$. This general pattern of behavior occurred for both IB1 and IB2 in all the applications. There is some optimal setting for k at which IB1 records its highest average classification accuracy. Its accuracy gradually tapers off with higher and lower settings. This occurs because predictions derived from a set of similar instances decrease the effects of noise, which most severely decreases performance for the $k = 1$ setting. However, if too many instances are used (i.e., if k is too high), then the average similarity of the instances used to generate predictions becomes lower. Since less similar instances less frequently have the same target value, they can be expected to yield less accurate predictions. This explains why IB1's average accuracy peaks and drops with increasing settings for k. This also occurred for the Waveform-22 application, where IB1's highest average accuracy (81.5%) was achieved when $k = 11$. In this case, IB1's accuracy will peak at higher values for k. This accuracy was 3.5% higher than IB3's best and a full 10.0% higher than C4's accuracy on this domain.

However, IB3 and IB4's learning behavior is extremely sensitive to k's setting. They performed best when $k = 1$. Their accuracies decreased quickly to random guess as k is

increased. This occurs because, as k increases, it becomes exceedingly difficult for instances to become acceptable. When k is large, dissimilar instances are used to generate prediction decisions, which are invariably wrong. This decreases their estimated accuracy and they are eventually eliminated from the partial concept description. This explains why IB3 and IB4's storage requirements decreased quickly in Figure 5.5. This also occurred for the LED-7 and Waveform-21 applications. Their standard errors increase during this time before settling to lower values for higher values of k. This occurs because IB3 and IB4 will occasionally manage to accept instances for low values of $k > 1$, which will allow them to record high accuracies in learning trials. However, when k is large, then IB3 and IB4 consistently record predictive accuracies near chance. The standard errors are lower for these settings.

These results suggest that IB3 cannot work well with higher values of k. However, it can be modified to improve its performance for these settings. For example, it can be trained using $k = 1$ and then be directed to use $k > 1$ settings during testing. Lower initial values for acceptance will also improve IB3 and IB4's learning behavior. These modifications have not yet been evaluated and they remain a project for future research.

Databases with Imperfect Attribute Sets:

As shown in the figure, the IBL algorithm's behavior for the Cleveland database application was similar to their behavior for the applications to the noisy artificial domains. IB1's highest accuracy (80.4% at $k = 5$) was again higher than IB3 and IB4's best (78.4% and 78.1% respectively). It was also 5.2% higher than C4's average accuracy. Also, IB1's accuracy continually improved for the Hungarian database application, beginning at 56.1% and ending at 71.6%. These results suggest that, although the majority of similar instances have similar classifications in this database, more than 1 instance is required to tolerate its large number of missing values, its imperfect attributes, and its noise to achieve high predictive accuracies. Its accuracy should approach IB3's and IB4's with sufficiently high values of k.

IB3 and IB4 again performed poorly. Their accuracies quickly dropped to chance as k was increased.

Noise-Free Databases

The IBL algorithms performed similarly in the Congressional Voting database application. However, this is the only application where IB1's apparent peak (92.9%) is lower than IB4's best (93.8%). IB2 performed relatively well in this application, recording accuracies as good as IB1's for higher values of k. This occurred because there is not much noise in this dataset. IB2's good performance was anticipated by the experiments in Section 4.2.4, which showed that IB2's accuracy was as good as IB1's in an application to an artificial domain with low levels of noise.

C4's accuracy (95.6%) was highest for this domain. It's possible that C4 will record higher classification accuracies than IBL algorithms for noise-free applications.

Noisy Domains with Irrelevant Attributes:

Higher values for k repair IB1's low accuracies for both the LED-24 Display and the Waveform-40 artificial domain applications. IB1's peak occurs for $k \geq 11$ for both of these applications. Its highest average accuracies for these domains (i.e., 67.2% and 76.5% respectively) are higher than the highest for IB3 and IB4. They are also higher than C4's accuracies, which were 66.9% and 70.9% respectively.

Summary:

IB1's behavior improved in each application as k was increased. Its accuracy peaks for a certain value of k and will, by definition, approach the frequency of the target value with the highest observed relative frequency as k increases to the size of the training set. IB2's behavior is similar, although its accuracies are slightly lower. IB3 and IB4's behavior is sensitive to the setting for k. They perform well when $k = 1$, but poorly when $k > 1$. Moreover, IB1's highest accuracy was better than IB3 and IB4's best in five of the seven applications. In the other two applications, it approached IB4's best for the Congressional Voting database and is still rising for the Hungarian heart disease database application. It can be argued that IB3 and IB4 cannot, in their current form, be relied on to yield higher classification accuracies than IB1. However, a search is required to find the setting for k which yields IB1's peak classification accuracy. Also, the other algorithms sharply reduce IB1's storage requirements. Finally, IB3 and IB4's performance can be improved by modifications that allow instances to be more easily accepted earlier in the training process. In summary, IB3 and IB4 should perform well when $k > 1$ after suitable modifications.

The IBL algorithms' behavior for numeric prediction tasks was somewhat similar to their behavior for symbolic prediction tasks. Table 5.9 summarizes their behavior for four of the seven numeric prediction tasks. Figure 5.6 summarizes the results for the heart disease diagnosis task using the Cleveland database. As with the symbolic prediction tasks, IB1 and IB2's accuracies increase with k, peaking at some value before degrading to a weighted-average guess using all the instances in the partial concept description. However, the performance of IB3 and IB4 is no longer always at an optimum for $k = 1$. Instead, the performance of these algorithms also follows a peaking curve, as shown in the figure. As before, IB3 and IB4's storage requirements decrease with higher values of k, indicating that the acceptance test is more difficult to pass when more instances are used to generate target value predictions. The following paragraphs describe the results in more detail.

Skewed Distributions:

The results for the degree of heart disease target function for the Hungarian database are similar to those shown for the Cleveland database. However, IB1's lowest average relative error, obtained when $k = 11$, is still high (0.33). The results suggest that it will decrease more with increasing values of k, but IB1's "recovery" from this database's array of challenges is slower than it was for the same symbolic prediction task. IB3 and IB4's relative errors dip

Table 5.9: Average relative error and storage requirements for three of the numeric prediction tasks.

Target Function	k	IB1	IB2	IB3	IB4
Tumor Size	1	0.22	0.23 62.0	0.21 16.1	0.21 17.5
	3	0.20	0.20 55.8	0.19 17.8	0.19 17.6
	5	0.19	0.20 53.8	0.18 18.1	0.19 17.6
	7	0.19	0.19 53.0	0.18 17.6	0.18 18.0
	9	0.19	0.19 51.8	0.18 17.3	0.18 18.2
	11	0.19	0.19 51.8	0.17 18.3	0.18 18.4
Horsepower	1	0.04	0.06 24.9	0.08 16.8	0.07 17.8
	3	0.05	0.07 20.8	0.07 16.3	0.07 17.1
	5	0.05	0.07 21.4	0.08 15.8	0.08 16.6
	7	0.05	0.07 21.7	0.08 16.1	0.08 16.6
	9	0.06	0.07 22.5	0.09 13.7	0.09 13.3
	11	0.06	0.07 23.7	0.10 12.2	0.11 9.9
# of Months	1	0.20	0.22 57.4	0.20 18.4	0.20 17.3
	3	0.17	0.18 55.4	0.17 20.7	0.18 20.7
	5	0.17	0.18 55.0	0.16 23.3	0.17 22.3
	7	0.16	0.17 54.8	0.17 21.7	0.17 22.4
	9	0.16	0.17 55.8	0.20 18.9	0.19 18.3
	11	0.16	0.17 56.1	0.22 13.9	0.22 13.0

to approximately 20% at $k = 5$ before rising. Their storage requirements also peak at this setting for k.

The results with the horsepower prediction function are unique due to its low level of noise. In this case, all the algorithms peaked for the $k = 1$ setting. Storage requirements for the IB3 and IB4 algorithms immediately decreased with increasing values for k.

Normal Distributions:

IB1 and IB3 tied for the lowest relative error when predicting the number of months lived after a heart attack. However, IB3 peaked at $k = 5$ again while IB1's lowest average error was recorded at $k = 11$, the highest value for k tested. The patterns for the algorithms' behavior in this experiment were similar to the other experiments.

The patterns repeat again for the cholesterol level prediction task. However, IB3 and IB4's peaks occur when $k = 9$. IB2's storage requirements decrease gradually, indicating that it makes fewer errors when it relies on larger numbers of instances for predicting target values.

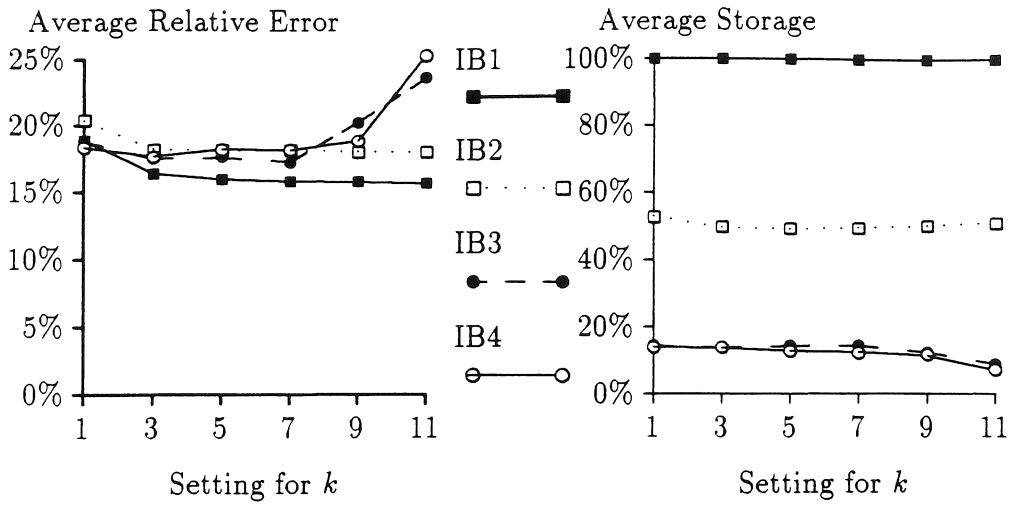


Figure 5.6: Average relative error and storage requirements for the degree of heart disease prediction task (Cleveland database) as k is varied.

Finally, the results for predicting breast cancer tumor size all have peaks at $k \geq 11$. IB3's predictive accuracy (17.5% average relative error) is about equal to average guess's relative error (17.1%) and is still better than IB1's (18.8%) at this setting. Performance improvement with increasing values of k is obviously slowing down between the two highest values for k tested. The familiar peaking pattern should occur for slightly higher settings for k.

Summary:

The results for the numeric prediction tasks were relatively consistent across the six applications tested. The performance of the algorithms peaks at a certain value for k and decreases gradually afterwards. IB3 and IB4 are not as sensitive to this parameter's setting for numeric prediction tasks as they were for symbolic prediction tasks. This occurs because, for numeric prediction tasks, most-similar instance predictions are rarely better than a weighted average prediction generated from a small set of highly similar instances.

The setting for k at peak performance varied from 1 for the horsepower target function to at least 11 for the prediction of tumor size. A brief analysis of the applications shows that these differences reflect the sparsity of the application's instance spaces. Table 5.10 displays the value of k for IB3's peak performance in each application and a rough estimate of the sparsity of each application (i.e., the dimensionality of the database divided by the size of its training set). The peak occurs at lower settings for k for sparse instance spaces and higher settings for more densely populated spaces. It's not surprising that the horsepower target function, where the peak performances occurred when $k = 1$, has the largest sparsity

Table 5.10: The value of k that supports IB3's peak performance increases with the sparsity of the application, estimated by the size of the training set divided by its dimensionality.

Database Application	Sparsity Measure	Peak for IB3
Horsepower	17.4	$k = 1$
Number of Months	9.8	$k = 5$
Hungarian Degree of Heart Disease	6.3	$k = 5$
Cleveland Degree of Heart Disease	6.1	$k = 7$
Serum Cholesterol Level	6.1	$k = 9$
Tumor Size	5.0	$k = 11$

measure. This measure can be used to estimate which setting for k should be used for IB3 in numeric prediction tasks.

5.3 Alternative Learning Components

This section describes a high-level parameter study for the IBL framework's learning component, which consists of the memory updating function. The purpose of this study is to determine whether a popular alternative to the design choice made for the learning component supports significantly improved learning behavior in the types of database applications studied throughout this dissertation.

The sequence of IBL algorithms described in Chapter 4 are examples of the simplest family of IBL algorithms, namely *instance-filtering* learning algorithms. These algorithms do not modify the set of instances selected to be saved, although their interpretations are modified as instance accuracy and relative attribute relevance information is learned.

The most popular alternative to instance-filtering IBL algorithms are *instance-averaging* IBL algorithms. These algorithms differ only in how they process instances whose target values are correctly predicted. Instead of discarding them, they are averaged with the instance responsible for the prediction.

There are different ways to average two instance's numeric attribute values. For example, Kohonen (1986) suggested using a cautious averaging strategy in which the classifying instance was replaced by an instance that was only $1/5^{th}$ of the distance towards the instance that was correctly classified. This fraction is slowly reduced over time. Although this strategy has its advantages, Kohonen embedded it in a non-incremental learning algorithm that introduced several additional parameters that require tuning by the user. Therefore, I chose to investigate Bradshaw's (1987) Disjunctive Spanning algorithm, which does not introduce additional parameters and is more familiar to the machine learning community.

The Disjunctive Spanning algorithm is an instance-averaging IBL algorithm that replaces an instance's values using a weighted-average of the two instance's values. Weights are assigned to each instance, they are initialized to one, and they are incremented by one each time the corresponding instance is averaged. The weighted-averaging approach ensures that the affects of the averaging process are reduced over time. That is, the stored instance will be "moved" successively smaller distances in the instance space with successive averagings. An instance with a large weight behaves like an instance saved by an instance-filtering algorithm because its values will be virtually unchanged with subsequent averagings.

This suggests that instance-averaging and instance-filtering algorithms will behave similarly after several averagings have taken place. If the target concepts are describable by a prototype plus noise, then instance-averaging algorithms could support superior learning behavior since the averagings help to filter the noise. For example, the disjunctive spanning algorithm should perform well on the LED Display and Waveform domains. However, these algorithms can yield an averaged "instance" that (1) does not exist due to constraints in the application domain or (2) has a different target value. The latter situation occurs when the set of instances along the line segment (in the instance space) between the two instances to be averaged have different target values. For example, the instances being averaged may lie in different disjuncts. If this situation occurs often, then the disjunctive spanning algorithm's predictive accuracy will decrease.

I applied variants of the disjunctive spanning algorithm to the set of symbolic prediction tasks introduced in Chapter 4 to gather information on their learning behavior. The results are summarized in Table 5.11. IB1's results are not included in this table since it simply saves all training instances. IB2, IB3, and IB4 were modified to average correctly classified instance instances rather than filter them. I allowed the disjunctive spanning variants to interpret Boolean/binary-valued attributes as numeric-valued, but did not implement any notion of averaging for symbolic values. The results, which are discussed in detail in the following paragraphs, confirmed most of my expectations.

Noisy Artificial Domains:

The disjunctive spanning (DS) algorithm performed well in the Waveform-21 application because the target concepts are prototypes plus added noise. Instance-averaging helps to eliminate this noise. IB2's improvement in accuracy was significant ($t(24) = 1.89, p < 0.05$). However, DS did not perform particularly well in the LED-7 display domain even though it also has these domain characteristics. This was due to the small number of training instances (200) in this application and the large number of target concepts (10). Since approximately 50% of the instances in this domain are noisy, few averagings took place during each learning trial. Thus, the instance-averaging algorithms' learning rates are not faster than the instance filtering algorithms in this application.

Table 5.11: Average percent accuracy ± standard error and percent storage requirements for the instance-filtering and instance-averaging algorithms.

Database	Method	IB2	IB3	IB4
LED-7	Filtering	63.0±0.9 41.6	72.5±0.4 20.1	32.1±3.0 21.3
	Averaging	61.6±0.9 43.1	70.3±0.6 20.0	19.4±3.0 17.1
Waveform-21	Filtering	68.4±1.1 32.3	74.2±1.2 11.1	76.9±1.0 13.4
	Averaging	77.6±0.8 38.8	80.3±0.7 11.6	80.2±0.9 14.6
Cleveland	Filtering	71.4±0.9 32.0	78.4±0.9 3.9	78.1±0.7 4.2
	Averaging	72.9±0.8 28.6	78.1±1.7 3.5	78.1±0.9 3.7
Hungarian	Filtering	53.1±2.4 36.9	79.4±0.9 4.3	79.8±0.7 3.9
	Averaging	55.8±2.2 34.5	78.0±1.4 3.7	79.5±1.4 3.5
Voting	Filtering	90.9±0.5 11.6	90.6±0.6 3.5	93.8±0.4 3.0
	Averaging	89.3±0.7 14.2	89.1±0.8 4.7	93.2±0.5 3.8
LED-24	Filtering	43.7±0.8 60.1	46.6±0.7 25.3	66.1±0.6 25.4
	Averaging	49.8±0.7 54.8	56.6±0.7 17.1	73.2±0.7 15.2
Waveform-40	Filtering	64.0±0.7 38.3	67.2±1.1 11.8	72.1±1.2 12.6
	Averaging	73.3±1.0 30.0	73.5±1.0 5.6	73.4±1.1 6.5

Databases with Imperfect Attributes:

The algorithms performed equally well when applied to the two heart disease databases. The instance-averaging algorithm could not repair IB2's problems with the latter database, indicating that, like the instance-filtering variant of IB2, it suffers from the wrong (1) choice of normalization function, (2) algorithm for processing missing attribute values, and (3) setting for k for this application. Since the instance-averaging algorithms had low storage requirements for IB3 and IB4, the results indicate that the predictor attributes are imperfect or that the target concepts are not describable only by prototypes. Otherwise, the disjunctive spanning variants would have recorded near-perfect classification accuracies.

Noise-Free Databases:

The instance-averaging algorithms did not outperform the instance-filtering algorithms in the application to the Congressional Voting database. They recorded similar average classification accuracies and slightly higher storage requirements. The instance-averaging variant for IB4 was again able to learn attribute relevance information that resulted in increased accuracies.

Applications with Irrelevant Attributes:

The instance-averaging algorithms performed exceptionally well on the LED-24 and Waveform-40 applications. The latter was expected for the same reasons that they performed well for the Waveform-21 application. The improvement in IB2's accuracy was again

significant ($t(24) = 1.85, p < 0.05$). There were also large reductions in storage requirements for all three algorithms, indicating that the instance-averaging variants learned prototypes of the three target concepts.

The instance-averaging algorithms' most spectacular improvement was with IB4 in the LED-24 Display domain. IB4's instance-averaging accuracy increased significantly ($t(24) = 1.87, p < 0.05$). Furthermore, its average accuracy (73.2%) was close to the 74% Bayes optimal rate and well above C4's average accuracy (66.9%). It also substantially reduced storage requirements for all three IBL algorithms. Instance-averaging learning components perform well in applications where the target concepts are defined by prototypes and the noisy training instances are described by irrelevant attributes.

Summary:

In summary, the instance-averaging algorithms perform about as well as the instance-filtering algorithms except when the target concepts are defined by prototypes and when the instances are described by multiple irrelevant attributes. Under these conditions, several of the algorithms using an instance-averaging memory updating function recorded significantly increased classification accuracies and significantly decreased storage requirements.

Bradshaw (1985) used the disjunctive spanning algorithm in his NEXUS speech recognition architecture. He compared its performance (i.e., classification accuracy) against CICADA, a non-learning speech recognition system developed at Carnegie Mellon University, on a two-speaker letter-pronunciation database. NEXUS was able to correctly recognize the speakers' letter in an average of 93% of the classification attempts after having cycled through the database thirty times. CICADA's accuracy was 80%. One reason that NEXUS performed well is because the concepts are defined by prototypes. This is an excellent application for the disjunctive spanning Algorithm.

Bradshaw didn't evaluate NEXUS versus other learning algorithms. Based on the results in this section and Aha and Kibler's (1989) evaluations, it appears that this algorithm performs well in comparisons with decision tree algorithms in a large set of applications.

I also applied the disjunctive spanning algorithm to the standard set of six numeric prediction tasks. The results are listed in Table 5.12 In summary, the instance-averaging memory updating function did not improve learning behavior for these tasks. Average relative error rates and storage requirements were roughly identical. This suggests that the target concepts in these applications, unlike in the artificial LED and Waveform domains, are not accurately describable with a single prototype. Furthermore, since the experiments in Section 4.4.3 showed that the instances in these database applications have few irrelevant attributes, it is not surprising that the instance-averaging variants' did not outperform the instance-filtering algorithms.

Table 5.12: Average relative error and and percent storage requirements for the instance-filtering and instance-averaging algorithms.

Target Function	Method	IB2	IB3	IB4
Tumor Size	Filtering	0.23 62.0	0.21 16.1	0.21 16.6
	Averaging	0.23 62.5	0.21 16.5	0.21 15.6
Cholesterol Level	Filtering	0.15 52.3	0.13 13.6	0.14 13.3
	Averaging	0.15 51.7	0.14 13.2	0.13 13.1
Degree of Disease (Cleveland)	Filtering	0.20 52.9	0.19 14.2	0.18 13.9
	Averaging	0.20 52.4	0.18 12.3	0.17 12.7
Degree of Disease (Hungarian)	Filtering	0.43 36.7	0.23 6.3	0.20 7.5
	Averaging	0.44 33.8	0.22 7.1	0.21 7.0
Horsepower	Filtering	0.06 24.9	0.08 16.8	0.07 16.0
	Averaging	0.06 24.7	0.08 17.6	0.07 16.1
Number of Months to Live	Filtering	0.22 57.4	0.20 18.4	0.20 17.3
	Averaging	0.24 60.7	0.23 17.9	0.22 17.4

5.4 Chapter Summary

This chapter described five investigations with alternative design choices for IBL algorithms. These high-level parameter studies are unique; they have not appeared in other publications concerning IBL algorithms. Therefore, it is not surprising that several the conclusions drawn from these studies offer several contributions to the literature on instance-based learning algorithms:

1. Pre-processing component: *The choice of normalization function can significantly alter learning rates and classification accuracies.* Surprisingly, the linear normalization method performed relatively well in comparison with the standard normalization algorithm. However, normalization should be done; the no-normalization option worked well only when the attribute ranges corresponded to relative attribute relevance.
2. Performance component:
   (a) Similarity function: *The choice of algorithm for processing missing attribute values can significantly alter learning rates and classification accuracies.* None of the three strategies tested consistently outperformed the others.

   Furthermore, *metrics for processing symbolic attribute values that compare constraints on target value distributions do not always outperform the simple overlap metric.* In fact, no clear consensus emerged from these experiments, suggesting that further study is needed in this area.
   (b) Prediction function: *The performance of IB1 and IB2 generally improve with increasing values of k. The level of IB3 and IB4's performance both decrease with*

increasing values of k. Furthermore, there appears to be a best value of k for IB1 and IB2 in each application that can be approximated by analyzing simple domain characteristics such as dimensionality and training set size. IB3 and IB4 should be extended to relax their high instance-acceptance standards during the early parts of the training process so that they can perform well when $k > 1$.

3. Learning component: *Instance-averaging algorithms can support significantly improved learning behavior in application where the target concepts are described by prototypes and where the instances are described by multiple irrelevant attributes.* However, these algorithms otherwise perform at about the same level as instance-filtering algorithms and are susceptible to deriving "instances" in their partial concept descriptions that are misclassified.

The purpose of this chapter was to begin to gain insight concerning the quality of the design decisions used for the basic IBL algorithms introduced in Chapter 4. The investigations suggested that alternatives to the basic algorithms should be explored and that more investigations (e.g., factorial designs) and domain characterization analyses are required to determine when each alternative should be used.

Several other alternatives and design decisions should be explored that were not discussed in this chapter. For example, a normalization function that normalized using a geometric averaging function may lead to improved learning behavior. So may alternative similarity functions, such as the city-block distance function, which Cost and Salzberg (1990) claim works as well in PEBLS as does the Euclidean distance function. Similarly, similarity functions that decrease exponentially with the distance between pairs of instances should also be investigated since there is a great deal of empirical evidence that people use these functions (Shepard, 1987). These functions are investigated in Chapter 6. Another alternative not discussed in this chapter are prediction functions for learning graded concept descriptions (Aha, 1989a). IBL algorithms can learn such descriptions, but the method by which they do this has not yet been analyzed. Finally, alternative memory updating functions have not been fully analyzed. For example, one of Bradshaw's (1987) most pressing concerns with the Disjunctive Spanning algorithm is that it has a tendency to allow classifications to take place between instances that are distant in the instance space. While investigations have examined adding distance-thresholding methods to instance-filtering and instance-averaging algorithms (Kibler & Aha, 1988), no investigations have compared and analyzed alternative means for solving this problem. The results of these investigations may suggest that alternative instance-averaging algorithms could solve this problem, such as Kohonen's (1986) algorithm, which cautiously averages instances instead of replacing them by their weighted average.

In conclusion, this chapter only partially investigated the IBL algorithm design choices and left more questions unanswered than solved. Nevertheless, this is a sign of progress. These investigations now suggest which design decisions are critical and should be the focus of future studies.

Chapter Acknowledgements

John Gennari, Pat Langley, and Dennis Kibler provided guidance on organizing the empirical analyses in this chapter. In fact, John's (Gennari, 1990) partitioning of empirical analyses in his Chapters 4 and 5 parallel the organization of the same-numbered chapters in my dissertation. Dennis, Steven Salzberg, and Gary Bradshaw suggested further alternatives to study after I had completed this chapter, but there comes a point when enough is enough! There are an infinite number of possible alternatives for instantiating the IBL framework. I chose to examine a lesion and an intuitive alternative in each study. Hopefully, the examinations of these alternatives have helped us to extend our understanding of these algorithm's capabilities.

Chapter 6
Psychological Analyses

Gaining a deeper understanding of human learning will continue to provide important clues about what to imitate and what to avoid in machine learning programs...it follows that among the most important kinds of learning research to carry out in AI are those that are oriented towards understanding human learning.
– Herbert Simon (1983, page 36)

This...is exactly the way of cognitive science – moving freely back and forth between the data-oriented methods of psychology and the system-oriented methods of artificial intelligence.
Alan Newell (Laird, Rosenbloom, & Newell, 1986, page xiii)

Exemplar models in general tend to be short on processing details.
– (Medin & Shoben, 1988, page 184)

Research in artificial intelligence (AI) has often influenced the development of psychologically plausible theories in the cognitive science literature, dating as least from the time that Newell, Shaw, and Simon's (1963) observations that search plays a central role in AI systems was also found to be true for human problem solving behavior (Newell & Simon, 1972). Similarly, several psychologists and computer scientists (e.g., Anderson, 1983; Fisher, 1987; Schlimmer, 1987a; Langley *et al.*, 1989) have used the conclusions drawn from experiments with cognitively plausible models to constrain their computational models. As a burgeoning branch of artificial intelligence, machine learning is no exception to this ongoing cooperation between these disciplines. More specifically, several instance-based learning systems were inspired by psychologically plausible models of categorization. Therefore, any discussion on instance-based learning algorithms would be incomplete without a description of *exemplar-based* models of categorization, which assume that predictions are based solely on the dynamic extraction of information from a set of stored exemplars.

However, this chapter contains more than an introductory survey on exemplar-based models of categorization; it also contains the first contribution to the literature on these models from a computer scientist's perspective. My contribution centers on demonstrating that an instance-based learning algorithm can improve current exemplar-based models'

fits to subject data. This is important since, as Medin and Shoben (1988) have noted, exemplar-based models are seldom if ever fully specified in their literature. In this regard, the contributions in this chapter complete the loop referred to above by Newell: the algorithms described in Chapter 4, whose study advanced the understanding of instance-based learning algorithms in the machine learning literature, were inspired by models in the categorization literature. After much elaboration, I have now used them to extend the body of knowledge on psychologically plausible exemplar-based models.

This chapter reads a bit differently than the others because the vocabulary of cognitive psychology differs somewhat from that in computer science. I will use the term *experiment* to denote a study with biological subjects and *simulation* to denote an investigation with a computer program. *Feature* will be used synonymously with *attribute-value pair*. The common phrase *exemplar-based model* will be used to refer to psychologically plausible instance-based algorithms for categorization. However, models are not necessarily equitable with algorithms since they are generally left underspecified. A fully specified model of categorization is called a *process model*. The term *exemplar* will be used to denote a specific instance even though Smith and Medin (1981) allowed it to also denote a subset of a category (e.g., the exemplar *robins* is a subset of the category *birds*).

The models described in this chapter address only symbolic prediction tasks. Although categorization models are now being developed for predicting numeric values, the vast majority of work on models of categorization addresses the modeling of symbolic concepts. Also, these models ignore many issues of interest to the machine learning community, such as missing values and noisy data.

This chapter begins in Section 6.1 with a discussion of the motivation for attending to the study of psychologically plausible models of categorization. The research related to the development of exemplar-based models is surveyed in Section 6.3. Four exemplar-based process models (i.e., IBL algorithms) are introduced in Section 6.4, evaluated against each other in simulations in Section 6.5, and are evaluated against subject data in Section 6.6, which describes an experiment with an artificial domain that investigates whether subjects' categorization processes are sensitive to context. The results show that the most context-sensitive of the four IBL algorithms also provides the best fits to subject data. Section 6.7 completes the study of exemplar-based models with a discussion concerning how these results fit with other studies on psychologically plausible models concerned with context sensitivity, including several recent proposals concerning context-sensitive exemplar-based models. Benefits and limitations of exemplar-based models are also described. Finally, Section 6.8 summarizes the contributions of this chapter.

6.1 Motivation for Studying Cognitive Models

There are several reasons for studying the literature on psychological models of categorization. The reason most relevant to this discussion is that they provide behavioral guidelines for computational models of learning. Several learning systems have been designed to display psychologically plausible behavior, including SOAR (Laird *et al.* 1986), PUPS (Anderson, 1989), and ICARUS (Langley *et al.*, 1989). Under the assumption that intelligence can be computationally simulated, these three integrated cognitive architectures attempt to computationally model (among other things) psychologically plausible learning processes For example, SOAR's central thesis is the *chunking theory of learning*, which proposes that practice improves performance via the acquisition of knowledge about patterns, called *chunks*, from the task environment. Their theory was developed from evidence that chunking accounts for some aspects of human problem-solving behavior (Miller, 1956; Bower, 1972; Chase & Simon, 1973). PUPS, which embodies another theory of how production rules are acquired, uses *spreading activation* to implement its method for processing information. Several researchers have described evidence showing that spreading activation methods account for some aspects in human problem-solving behavior (e.g., Collins & Quillian, 1972; Meyer & Schvaneveldt, 1971; Anderson, 1974). Finally, ICARUS's architecture, which is based on the development of a *probabilistic concept hierarchy*, was directly influenced by Smith and Medin's (1981) discussion on the psychological plausibility of probabilistic concepts. In summary, these three systems are applicable to a large set of problem solving tasks and their generality is derived, at least in part, from their concern with modeling cognitively plausible behavior.

None of these systems are specifically instance-based, although two of them (PUPS and ICARUS) retain and exploit specific instance information during problem-solving episodes. Although several learning systems that use an instance-based approach have little or no connections with the psychological literature (e.g., Bradshaw, 1985; Stanfill & Waltz, 1986; Kurtzberg, 1987; Connell & Utgoff, 1987; Clark, 1989) research on psychologically plausible models of categorization directly inspired the development of several other IBL systems (e.g., Bareiss, Porter, & Wier, 1987; Kibler & Aha, 1987; Salzberg, 1988). In fact, all three groups derived their motivation from Smith and Medin's (1981) discussion on exemplar-based models. Each group reported that IBL algorithms support surprisingly good computational learning behavior in practical supervised learning applications. Thus, psychological research on modeling categories directly inspired the development of well-behaved learning algorithms.

Although Bareiss (1989) and Salzberg's (1990) dissertations were inspired by exemplar-based models of categorization and lead to the development of interesting computational learning algorithms, they did not include contributions to the psychological literature. This is unfortunate since many of the ideas they developed with their colleagues and used in their algorithms could extend the capabilities of psychologically plausible exemplar-based models.

Section 6.6 contains an evaluation of the psychological plausibility of an exemplar-based process model that was inspired by the design of IB4, a computational learning algorithm with no claims of consistency with known psychological behavior. More specifically, Section 6.4, describes an investigation of the fit of four IBL algorithms to data collected from subjects. The learning task involves a simple artificial domain that encourages using a context-specific attribute weighting strategy. The objective of the experiment is to show that subjects use contextual information to determine an attribute's relevance in prediction tasks. This is a novel experiment; this is the first investigation showing that context-sensitive prediction strategies improve the fits of exemplar-based process models.

6.2 On Dimensions for Distinguishing Psychological Models

It is necessary to survey the literature on psychological models of categorization to understand the background for the design decisions used in the creation of the exemplar-based process models described in Section 6.4. This section sets the stage for the survey in Section 6.3 by presenting three ways in which models of categorization can be differentiated.

There are many ways to characterize the ability of a model to account for psychologically plausible behavior. A good method for characterizing models should be lucid and based on a small number of easily-measured dimensions. Section 6.2.1 begins by describing the most popular method and explains why it is insufficient. An alternative method that was recently advocated by Barsalou (1989) is presented in Section 6.2.2 which also explains why his method is also insufficient. I introduce a new method for accomplishing this task in Section 6.2.3 and use it in the remainder of this chapter to compare and contrast models of categorization.

6.2.1 Category Structure

Smith and Medin's (1981) partitioning of models into three disjoint *views* is currently the most popular dimension for distinguishing models of categorization. However, as Table 6.1 shows, this dimension is in fact itself multi-dimensional. The following paragraphs summarize these three views.

Categorization is the process of determining that a specific instance is a member of a concept or that a concept is a subset of another. The primary purpose of studying categorization in humans is to develop an understanding of what behaviors are exhibited by humans during categorization tasks. A secondary purpose is to develop models whose behavior is

Table 6.1: The three views of categorization, summarized by the constraints they impose on attributes in concept descriptions and the choice of representation for concept descriptions.

Name	Constraints on Attributes	Representation
Classical	Necessary and sufficient	Centralized, conjunctive
Probabilistic	Probabilistic	Centralized, implicitly disjunctive
Exemplar	Describe at least 1 member	Distributed, explicitly disjunctive

consistent with known subject behavior. The central assumption of cognitive psychologists is that categorization plays an essential role in the majority of cognitive behavior.

Early researchers on category learning (e.g., Bruner, Goodnow, & Austin, 1956) suggested that people abstract relatively rigid summary descriptions for categories that were defined by necessary and sufficient features. Concepts were restricted to conjunctive descriptions and concept members were assumed to be equally representative of their category. Furthermore, the set of defining features for a category was assumed to be a proper subset of the corresponding set for its subordinate concepts. This was called the *classical view* by Smith and Medin (1981). Mitchell's (1977) *candidate elimination* algorithm, which locates conjunctive concepts using a version space approach, is perhaps the best-known machine learning algorithm that adopts the classical view.

This view fell from favor as more information became known concerning human categorization behavior. The following list highlights several properties displayed during human categorization behavior that are in opposition to the pure classical view.

1. Concepts are frequently disjunctive.
2. Unclear cases frequently arise during categorization attempts.
3. People cannot specify defining features for most concepts.
4. People tend to use non-necessary features to define concepts.
5. Concept members vary in their *typicality*, which is correlated with *family resemblance*, defined as an increasing function of the number of attributes in common with other members of the category and a decreasing function of this number for non-members (Rosch & Mervis, 1975; Rosch, 1978).
6. Concepts are often judged to be more similar to distant than to immediate subordinate concepts.

A few of the typicality effects exhibited by subjects are summarized in the following list.

1. The time required to classify an instance is an inverse function of its typicality.
2. Categorizations for typical instances are generally learned earlier than for other instances.

3. Most prototypical instances are given first (and most frequently) by subjects when asked to list members of a category.
4. Performance on the categorization of prototypes, even if not presented during training, decreases less than performance with other training exemplars when subjects are delayed before being tested.

Several attempts have focused on revising the classical view to account for these observations, but they have met with limited success.

Consequently, two other views have been developed that successfully model all of these behaviors. These alternative views comprehensively relax the definition of a concept, as shown in Table 6.1. The *probabilistic view* (Smith & Medin, 1981) extends the classical view by allowing attributes to have a probabilistic rather than a Boolean relationship to concept membership. This view allows concepts to be less stable; instances aren't guaranteed to be either members or non-members. Otherwise, the probabilistic view is similar to the classical view in that it also assumes that concepts are represented by a single *abstraction* derived during training. The probabilistic view allows for degrees of disjunctiveness by permitting different combinations of features to achieve above-threshold requirements for concept membership. It also defines unclear cases as those that don't quite reach this threshold, explains the lack of defining features and use of non-necessary features by requiring the latter in concept descriptions, and explains typicality effects by relating the similarity of a member instance's properties to its parent concept. Finally, the probabilistic view does not restrict concepts to have more properties in common with immediate than with distant subordinates. Several machine learning algorithms adopt the probabilistic view, including COBWEB (Fisher, 1987), STAGGER (Schlimmer, 1987a), Quinlan's (1990a) probabilistic extension for decision tree algorithms, Michalski's (*et al.*, 1986) probabilistic extension for the AQ series of algorithms, and Rendell's (1986) PLS1 algorithm.

The *exemplar view* takes a different tact; it assumes that categorizations are based solely on specific instance information. This view postulates that abstractions are derived at the time of retrieval and are not explicitly maintained. Thus, this view further relaxes the definition of a concept. The exemplar view prescribes *lazy* models of categorization; it does not advocate that people derive abstractions when exemplars are presented. Instead, it assumes that people based their categorization decisions solely from stored instance information. This view is explicitly disjunctive, explains unclear cases as instances having comparable summed similarities to multiple concepts, places no requirements on defining properties other than that they are represented by an exemplar (i.e., member), and explains typicality effects based on assumptions that concept exemplars have high family resemblance scores. Machine learning algorithm's based on this approach include Bareiss's (1989) Protos system, Salzberg's (1990) NGE algorithm, and Aha, Kibler, and Albert's (1991) instance-based learning algorithms.

6.2.2 Information Storage

Smith and Medin (1981) suggest that a mixed probabilistic and exemplar view is more psychologically plausible than either extreme view. As will be shown in later sections, this position has been hotly debated and has not yet subsided. There is currently no consensus on whether people use exemplar-only, abstraction-only, or mixed representations for categories, although several arguments have been made for each possibility. However, these contradictory (and, subsequently, distressing) arguments have at least had a common theme: if any argument for one of the three perspectives is published that discourages the plausibility of another, then surely a more elaborated model supporting the other perspective will soon be published that soundly refutes the original argument. Indeed, authors who now present favorable evidence for their models state explicitly that they do not rule out the possibility that other models, once properly extended, might improve fits. It appears that the only consensus in this literature is that it is impossible to determine which type of model subjects use to solve categorization tasks.

Barsalou (1989) is the strongest advocate of this perspective. He argues that trying to determine whether people use only exemplars or both exemplars and abstractions is futile because they are informationally equivalent. That is, abstraction-based approaches can contain information on specific instances (e.g., as in the property set model (Hayes-Roth & Hayes-Roth, 1977), information on attribute correlations, and can modify the abstractions over time. Likewise, exemplar-based approaches can be modified to resemble abstraction models. Thus the degree-of-abstraction dimension does not distinguish models of categorization. The focus, he argues, should be on comparing *representation-process* pairs, wherein both the model's representation and identification procedure (i.e., performance and learning component) must be considered to determine whether it supports psychologically plausible behaviors. In this sense, arguments rejecting types of representations is misguided. Since Smith and Medin's (1981) view-based dimension for distinguishing models is based mainly on representational issues and is lean on processing issues, their partition of categorization models into three views is misleading and, consequently, unsatisfactory for predicting a model's ability to exhibit psychologically plausible behavior.

Barsalou (1989) argued that process models *can* be distinguished by how they store information. He noted that exemplar-based representations tend to exhibit information duplication (i.e., stored instances can share some of the same attribute values) whereas abstraction-based models remove redundant information and tend to rely more on information revision (e.g., we will see in Section 6.3.2 that feature-frequency models maintain counts on attribute value occurrences, which must be incrementally revised as when new instances are processed). However, psychologically plausible abstraction-based models can exhibit both information duplication and contain redundant information (Hayes-Roth & Hayes-Roth, 1977) and plausible exemplar-based models exist that revise their instances

(Hintzman, 1986). Therefore, this method's ability to predict a model's consistency with psychologically plausible behavior is also flawed.

6.2.3 Ability to Utilize Attribute Correlation Information

Barsalou's argument was not in vain since he shifted the emphasis from representations to process models. I will use an alternative process-related dimension to distinguish models of categorization: *the degree to which the model utilizes information concerning inter-attribute correlations.* Medin and Schaffer (1978) originally introduced the distinction of models that use no correlational information, which they called *independent cue* models, and models that base their predictions on correlational information, which they called *relational coding* models. Independent cue models use an additive combination of attribute values to derive predictions whereas relational coding models derive predictions from an interactive combination of attribute-value information. Unlike Medin and Schaffer, I will highlight the continuous nature of this dimension and argue that it correlates better with a model's ability to simulate psychologically plausible behavior than the dimensions suggested by Smith and Medin or Barsalou.

Four methods for representing category information have been repeatedly studied in the psychological literature: prototype, feature frequency, exemplar-based, and connectionist networks. The following four sections describe models based on these representations and distinguish them based on their ability to utilize correlational information. I will highlight evidence for the hypothesis that *there is a positive correlation between the amount of attribute correlational information utilized by a psychological model and the number of psychological phenomena that they can explain.* The following sections will then introduce psychological models in increasing order of their ability to utilize attribute correlation information. These models will be evaluated on the basis of their relative learning speeds in simulations described in Section 6.5 and their ability to fit subject data in experiments described in Section 6.6. These evaluations will show that the models' abilities correlate with the amount of attribute correlation information that they utilize during the categorization process; the model that best utilizes attribute correlation information will perform best in the simulations and have significantly better fits to the collected subject data.

6.3 Psychological Models of Categorization

This section contains a summary of relevant research dating from the development of prototype models to Nosofsky's (1986) introduction of the generalized context model. The three models to be surveyed in this section represent categories by prototypes, feature frequency counts, exemplars, and by connectionist networks respectively. These representations can

be best introduced with an example. Suppose that the following three instances describing people are available for training:

Number	Weight	Height	Gender
Denise	Heavy	Average	Female
David	Average	Short	Male
Daniel	Skinny	Average	Female

If the target concept is *Female*, then the prototype model would represent the concept with a single prototype with values of *Average* for both weight and height. A simple feature frequency model would save counts on each attribute-value pair: weight[heavy]=1, weight[skinny]=1, weight[average]=0, height[tall]=0, height[average]=2, and height[short]=0. An exemplar-based model would simply represent the *female* concept by the set of saved instances, where *Denise* and *Daniel* would be interpreted as positive examples and *David* would be interpreted as a negative example.

6.3.1 Prototype Models

Posner and Keele (1968; 1970) popularized the *prototype model*, which represents a target concept with single prototype. A category prototype's location in instance space is the point that has minimal summed distance to all members of the category. Prototypes are commonly assumed to be learned from the incremental averaging of attribute-values. These probabilistic models are independent cue models because they categorize instances based on their distance to each category's prototype, where distance is defined as an independent summation of attribute-value differences. Posner and Keele described evidence that prototype models account for several of the behaviors listed on page 154, including (1) subjects abstract and retain prototypes when trained on random dot patterns, (1) subjects recognized prototypes more quickly and accurately than other instances, and (3) after a delay between learning and testing, their performance on learned patterns suffered even though they retained knowledge of the prototype. They also argued that abstraction occurs during learning rather than at the time of retrieval, which is contrary to the thesis of exemplar-based models.

Franks and Bransford (1971; Bransford & Franks, 1971) extended these results. They developed a prototype plus transformation model and found that categorization accuracy is inversely related to an instance's transformational distance to a prototype. They found no evidence that subjects based classification decisions on the memorization of specific instances in their experiments with geometric patterns. They showed that the fits of their model were significantly better than the fits with a simple *feature frequency* model, which determines categorization judgments based on the number of times a probe's features occurred in previously processed exemplars. More elaborate feature frequency models were not considered.

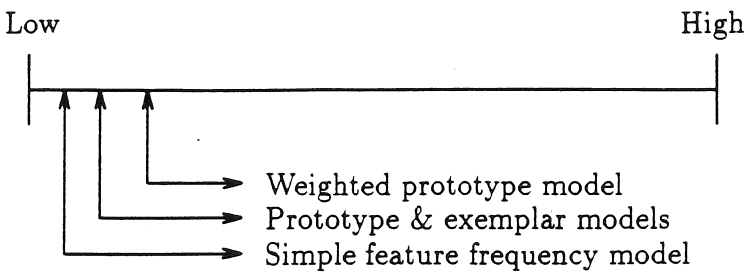


Figure 6.1: Early models' relative utilization of attribute correlation information.

Finally, Reed (1970; 1972) provided evidence that a weighted-prototype model often fit subject data significantly better than several other models, including a simple exemplar-based model (Sebestyen's (1962) *proximity* algorithm) that uses a nearest neighbor prediction algorithm. The weights on his models emphasized the relative relevance of features for predicting classifications. Reed also found that the proximity model fared better than his weighted-prototype model in one experiment, but he did not pursue the development of sophisticated exemplar-based models.

Figure 6.1 qualitatively describes the relative utilization of attribute correlational information by the four models mentioned in this section. The weighted-prototype model utilizes more correlational information than the primitive prototype model because its weights reflect the relative contributions of the attributes. Assuming that the target concept is describable by a single prototype, the prototype model is higher along this dimension than the simple feature frequency model, which completely discards correlational information in its representation and uses it only weakly during categorization. The proximity model retains all correlational information in its representation, but only uses some of it during processing (i.e., the information contained in the nearest instance). Therefore, it doesn't use correlational information as well as the weighted-prototype model.

Perhaps the most well-known limitations of "pure" prototype models is that they are insensitive to instance distributions. For example, this model will derive the same prototype for a concept whose only instance is the prototype itself and for a concept whose instances lie in a ring centered around the prototype. Although humans can learn distributions (Fried & Holyoak, 1984), the prototype model cannot distinguish these two concepts. Also, the pure prototype model, unlike people, cannot learn non-linearly separable categories (Medin and Schwanenflugel, 1981).

6.3.2 Feature Frequency Models

Neumann (1974) repeated Franks and Bransford's (1971) experiments and showed that a more elaborate feature frequency model fits their data more closely than does the prototype-plus-transformation model. His model differs from the simple one in that it also maintains higher-order feature frequency counts. That is, it maintains counts for pairs of features. Neumann called these *relational state frequencies.* This model's prediction function is the same for a first-order feature-frequency model: classify an instance as a member of a category c if all the frequency counts of relevant to the instance are higher for c than for any other category. Neumann's model was one of the first that was shown to be able to derive prototypes without maintaining them explicitly in memory. Higher-order feature frequency models can support this behavior because, as the distance of an instance to a prototype decreases, the presentation frequencies of its component frequencies increase sharply.

This increase can be exponential when counts are maintained on n-ary conjunctions of features. This is epitomized in the *property set* model (Hayes-Roth & Hayes-Roth, 1977), which maintains counts on the power set of feature conjunctions. This probabilistic-view model blurs the distinction between abstraction-based and exemplar-based models since it retains all specific instance information. Classification is based on the instance's most *diagnostic* frequency count, where the diagnosticity of a count for a category c is the count's value for c divided by the sum of its count for all known categories. Consider the instance with the two features *climate=sunny* and *air=smoggy.* Suppose that the category named *Costa del Sol cities* has feature counts *climate=sunny*[2], *air=smoggy*[0], and *climate=sunny*$\wedge$*air=smoggy*[0] and the category *New York cities* has the counts *climate=sunny*[1], *air=smoggy*[1], and *climate=sunny*$\wedge$*air=smoggy*[0]. The conjunctive feature count has no diagnosticity. The diagnosticity of the *climate=sunny* feature is $\frac{2}{3}$ for the Costa del Sol cities category and the diagnosticity of *air=smoggy* is $\frac{1}{1}$ for the New York cities category. Since this latter feature is most diagnostic, the instance would be classified in the New York cities category.

The relative measures for the models mentioned in this section are shown in Figure 6.2. The property-set model provided better fits of subject data from a simple artificial domain than 24 other models, including a prototype model, a simple feature frequency model, and the simple proximity exemplar-based model. This isn't surprising since the property-set model explicitly maintains information on all conjunctive relations of features. It also retains more correlational information than Neumann's (1974) feature-pairs model, which in turn makes better use of correlational information than does the proximity algorithm.

Hayes-Roth and Hayes-Roth (1977) concluded that their property-set model of categorization provides the best theoretical explanation available for classification performance at that time. However, this model has a few serious limitations. First, as mentioned earlier, the number of feature counts required increases exponentially with the number of features used to describe the training instances. This problem may be alleviated by discarding counts

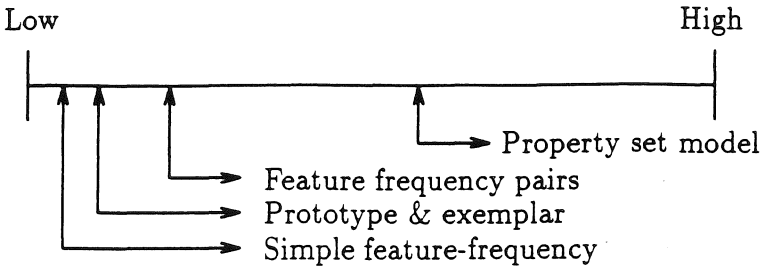


Figure 6.2: Feature frequency models' relative utilization of attribute correlation information.

for conjunctions that are not utilized during categorization tasks, as is done in both PUPS (Anderson, 1989) and STAGGER (Schlimmer, 1987a). A more serious limitation is that no feature frequency model has yet been developed to simulate selective attention processes, where some attributes are given more attention by people according to their utility in categorization tasks. Finally, these models have not been developed to work with numeric-valued attributes. Recent exemplar-based and connectionist models do not have these limitations.

6.3.3 Exemplar-Based Models

The Context Model

In their seminal paper, Medin and Schaffer (1978) introduced the *context model*, the first exemplar-based model whose similarity function is defined by a relational rather than an independent function of attribute-value comparisons. Their model states that the probability that instance x is a member of category t is the sum of x's similarities to all members of t divided by the sum of its similarities to all stored exemplars. That is,

$$\text{Probability}(x, t, P) = \frac{\sum_{y \in S(t)} \text{Similarity}(x, y, P)}{\sum_{z \in S} \text{Similarity}(x, z, P))}, \tag{6.1}$$

where $S(t)$ is the set of stored exemplars with target value t and S is the set of all stored exemplars. Similarity is defined as

$$\text{Similarity}(x, y, P) = \prod_{i \in P} \begin{pmatrix} 1 & \text{if } x_i = y_i \\ w_i & \text{otherwise} \end{pmatrix}, \tag{6.2}$$

where P is the set of predictor attributes and w_i is a pre-assigned weight for attribute i in the range $[0, 1]$. Low weights indicate that the predictor attribute is highly relevant for categorization tasks. The context model was unique among exemplar models; it was the first

Table 6.2: Training instances for the context model example.

Instance Number	Category	Attribute Values			
		a	b	c	d
1	A	1	1	1	1
2	A	1	1	1	0
3	A	0	0	0	1
4	B	0	0	0	0
5	B	0	0	1	1
6	B	1	1	0	0

model to (1) use all stored instances to derive similarity judgments, (2) use a multiplicative definition for similarity, and (3) define similarity without using a distance measure. The main benefit derived from using a multiplicative similarity function is to cause the similarity of two instances to decrease exponentially with their Euclidean distance. This allows "close" instances to have exponentially more influence on classification decisions than "distant" instances. Although this point was not emphasized in this way by Medin and Schaffer (1978), we will see that Nosofsky (1986) explicates it in Section 6.3.3.

An example should help to clarify how this model predicts target values. Suppose that the model is given the six training instances described by four binary-valued attributes (i.e., a,b,c, and d) for the binary classification task shown in Table 6.2. The exemplars of category A generally have more attributes with the value 1 than the exemplars of category B. The prototype for categories A and B are $\langle \frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3} \rangle$ and $\langle \frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{2}{3} \rangle$ respectively. Since the test instance x with attribute values $\langle a = 0, b = 0, c = 1, d = 1 \rangle$ is located midway between these two prototypes, a prototype model's classification would not favor either category. The property-set model (Hayes-Roth & Hayes-Roth, 1977) predicts x to be a member of category A because the most diagnostic features (both c and d) have a diagnosticity value of $\frac{2}{3}$ for category A. Both of these predictions seem somewhat absurd because the test instance is identical to category B's training instance number five. The context model predicts the probability that x will be classified in category A as

$$P(x \in A) = \frac{ab + abd + c}{ab + abd + c + cd + 1 + abcd} \tag{6.3}$$

and that it will be classified in category B as

$$P(x \in B) = \frac{cd + 1 + abcd}{ab + abd + c + cd + 1 + abcd} \tag{6.4}$$

where the values a, b, c, and d refer to the weights of the four attributes respectively. If we assume that these weights are equal and sum to 1, then $P(x \in A) = 23.5\%$ and $P(x \in B) = 76.5\%$, which clearly indicates that the context model predicts x to be a member of category B. This example highlights how the context model works: classification is most influenced by

the categorizations of similar instances, although slightly less similar instances also impact on prediction decisions.

Extensive testing has shown that the context model can account for several behaviors previously assumed accountable by only by abstraction-based models or mixed models, where both abstractions are maintained and specific instances are stored. For example, Medin and Schaffer (1978) showed that, like the higher-order feature frequency models, the context model can simulate prototypicality effects without maintaining explicit abstractions. The context model can simulate *differential forgetting* effects, where subjects tend to misclassify old exemplars more frequently than unseen prototypes, by selectively attending to attributes. Hintzman and Ludlam (1980) showed that exemplar-based process models can also simulate these effects by randomly deleting attributes over time. These two methods simulate a form of abstraction; specific instances are essentially replaced by partially remembered instances during categorization processes. The context model can also simulate subject behavior when applied to categories with large numbers of instances (Medin, Dewey, & Murphy, 1983), thus refuting arguments that both abstractions and specific instances are required to model this behavior (Robbins, Barresi, Compton, Furst, Russo, & Smith, 1978; Homa, Sterling, & Trepel, 1981). However, this is not to say that Medin and his colleagues believe that abstractions have no place in categorization models. In fact, Medin, Altom, and Murphy (1984) found that the type of information used (i.e., specific instances or abstractions) depends on the task; if subjects are trained on prototypes, then they are used during transfer tasks. However, if prototypes are not included among training instances, then they can be derived but are not necessarily stored by subjects.

The context model also explains subject data better than prototype models in many situations. For example, Medin and Schwanenflugel (1981) showed that independent cue models predict that linearly separable categories are easier to learn. They also found no evidence that humans learn linearly separable categories more easily than non-linearly separable categories. Not surprisingly, the context model provided significantly better fits than prototype models in several experiments when the target categories were not linearly separable. This is because the context model's thesis is that individual cases of high similarity between exemplars of different categories is the major determinant of task difficulty rather than linear separability. In another paper, Medin, Altom, Edelson, and Freko (1982) found that people are sensitive to correlated attributes. They described evidence showing that the context model provides better fits to subject data than prototype models, which are insensitive to correlated attributes because they derive predictions based on an *independent* summation of attribute-value differences.

Finally, Medin and Schaffer (1978) argued that the context model also provides better fits than feature frequency models because, in their current form, feature frequency models cannot differentially attend to features so that more relevant ones are emphasized during categorization tasks. In contrast, the context model's weights can be changed to simulate the selective attention to attributes during categorization tasks.

The MINERVA II Process Model

Although Medin and others showed that exemplar-based models can support several psychologically plausible behaviors, they never ruled out the possibility that subjects were using both derived abstractions and specific instances to solve categorization tasks. Indeed, Elio and Anderson (1981) explored this issue with an open mind when they evaluated the assumption that their ACT rule-based system requires saving only abstractions to accurately explain categorization behavior. Their experimental results convinced them that ACT should store specific instances and compute partial matches to stored production rules (i.e., abstractions). However, their claim that generalizations are abstracted and used for categorization tasks was rejected later by Hintzman (1984; 1986) in his simulations with his MINERVA II exemplar-based model.

Hintzman's model is worth discussing for several reasons. First, it is the most detailed exemplar-based process model in the psychological literature. Second, although it also represents categories by a set of stored instances, it embodies four processing assumptions that are radically different from those advocated by Medin and his colleagues. First, each presentation of an instance is assumed to leave a distinctive *trace* in memory. That is, if an instance is presented n times during training, then MINVERA II stores n copies of it in its partial concept description. This is done to help the model to explain how repetition affects behavior exhibited during the categorization process. Second, MINERVA II works like an associative memory; it can be used to predict any attribute's value. It inputs an instance described by a set of numeric values and outputs a result that denotes the average activation of each of the attribute values, any of which could be the target value of interest. Third, categorization is not a single-step process. Instead, this vector output is used as an input in the following time-step. This *echo resonance* procedure is repeated until the echo unambiguously resembles one of the stored category labels, which are uniquely encoded using a subset of the numeric-valued attributes. Finally, instances that are highly similar to a probe spread *activation* to similar instances. This is reminiscent of McClelland's (1981) node-network extension of the context model that spreads activation among stored instances and predicts target values based on which nodes have the highest activation levels.

MINERVA II linearly normalizes its numeric-valued predictor attributes to the range $[-1, 1]$, where 1 represents an excitation of the attribute and -1 represents its inhibition. Similarity between an instance x and a stored instance y is defined as the average of the product of their pairwise attribute values

$$\text{Similarity}(x, y, P) = \frac{\sum_{i \in P} x_i \times y_i}{|P|}, \tag{6.5}$$

where P is the set of predictor attributes. Hintzman decided to define similarity to decrease more quickly than linear with distance. He noted that this property is an important part of the context model.[1] Therefore, his categorization function is based on the *activation* of a

[1] I will discuss this point in detail in Section 6.3.3.

stored trace, defined as

$$\text{Activation}(x, y, P) = \text{Similarity}(x, y, P)^3 \tag{6.6}$$

The choice of exponent (i.e., 3) is based primarily on the fact that it preserves the sign of the similarity measure; Other odd-valued exponents would also suffice. Finally, the predicted value of any attribute i is defined as

$$\text{Feature\_prediction}(x, i, P, S) = \sum_{y \in |S|} \text{Activation}(x, y, P) \times y_i, \tag{6.7}$$

where S is the set of all stored instance traces. An instance's predicted categorization is the one that has the highest correlation with the target values of the echo resonance result.[2]

MINERVA II simulates a large number of psychologically plausible behaviors, including all of the prototypicality effects described earlier in this section. A few of its relevant characteristics are summarized in the following list.

1. Like humans, it learns concepts more easily when exemplars are low-level distortions of prototypes than when they are high-level distortions.
2. Learning is predicted to become more difficult as the prototypes of categories become more similar.
3. The MINERVA II model is similar to PUPS (Anderson, 1989) and SOAR (Laird, Rosenbloom, & Newell, 1986), which employ separate working and long-term memories. MINERVA II's long term memory is the set of stored traces. Its short term memory is the set of activations recorded for each of the attributes. However, MINVERA II is currently limited to encoding attributes using numeric values.
4. MINERVA II's echo resonance loop solves the ambiguous recall problem, where the initially output values for the target attribute(s) do not correspond well with any of the possible set of target values. The echo resonance loop improves correlation to the actual target value and was used to reject Elio and Anderson's (1981) argument that ACT needs to maintain explicit abstractions.

MINERVA II has only recently been evaluated on subject data (Hintzman, 1988). However, it has proven to be a qualitatively accurate simulator of many psychologically plausible behaviors. It assumes that instances are retrieved in parallel and that abstractions are implicitly derived from the summed responses of the traces that were most strongly activated. In summary, MINERVA II is a model that can simulate many different categorization behaviors without relying on stored abstractions. However, it relied on several elaborate extensions of the pure exemplar-based model to achieve these capabilities.

[2]MINERVA II encodes a category name using a *set* of target attributes.

The Generalized Context Model

Perhaps the most ambitious claim voiced recently in cognitive psychology is Roger Shepard's (1987) on his *universal law of generalization*, which states that the probability that two instances will be generalized (i.e., predicted to have the same target value) is an exponentially decreasing function of their psychological distance. A *psychological space* is an instance space with the following property: the probability that a response learned for an instance will generalize to similar instances is an increasing function of the distance between them. This law has extensive empirical support from studies with both pigeons and people over the past 35 years, both on artificial and real-world domains (e.g., spectral hues, shapes, morse code signals, vowel phonemes, etc.). Most of the recently proposed models of categorization exhibit behavior that is consistent with Shepard's law (Hayes-Roth & Hayes-Roth, 1977; Medin & Schaffer, 1978; Hintzman, 1984; Nosofsky, 1984; Gluck, Bower, & Hee, 1989; Aha & Goldstone, 1990; Kruschke, 1990).

Nosofsky (1984) was the first person to relate this empirical law to exemplar-based models when he showed that the context model's (Medin & Schaffer, 1978) multiplicative similarity function can be described as an instantiation of this universal law. Subsequently, Nosofsky (1986) developed the generalized context model (GCM), which explicates the relationship of the context model and Shepard's law as applied to the definition of similarity functions. The GCM defines similarity to decrease exponentially with distance

$$\text{Similarity}(x, y, P) = e^{-S \times \text{Distance}(x,y,P)^2}, \tag{6.8}$$

where distance is defined as a weighted variant of Euclidean distance

$$\text{Distance}(x, y, P) = \sqrt{\sum_{i \in P} w_i (x_i - y_i)^2}, \tag{6.9}$$

where P is the set of predictor attributes and the w_i are attribute weights of the form used in IB4. S is a *scaling* parameter that determines the degree of exponential gradient; higher values for S imply a steeper exponentially decreasing function of distance. Nosofsky showed that this definition of similarity is equivalent to the multiplicative similarity rule used in the context model:

$$\begin{aligned} \text{Similarity}(x, y, P) &= e^{-S \times \text{Distance}(x,y,P)^2} && (6.10) \\ &= e^{-S \sqrt{\sum_{i \in P} w_i (x_i - y_i)^2}^2} && (6.11) \\ &= e^{-S \sum_{i \in P} w_i (x_i - y_i)^2} && (6.12) \end{aligned}$$

Since

$$e^{-\sum_i f(i)} = \prod_i e^{-f(i)} \tag{6.13}$$

$$\ln e^{-\sum_i f(i)} = \ln \prod_i e^{-f(i)} \quad (6.14)$$

$$-\sum_i f(i) = -\sum_i f(i), \quad (6.15)$$

then

$$\text{Similarity}(x, y, P) = \prod_{i \in P} e^{-Sw_i(x_i - y_i)^2} \quad (6.16)$$

$$= \prod w_i' \quad (6.17)$$

where w_i' is attribute i's assigned weight in the context model and, when $x_i \neq y_i$, $w_i' = e^{-Sw_i(x_i - y_i)^2}$. The GCM uses the context model's definition for categorization, namely that an instance is classified according the category that maximizes Equation 6.1 on Page 160.[3]

The GCM model provided good fits in both in-depth studies and in experiments with larger numbers of subjects (Nosofsky, 1986; 1987; 1989). Two of Nosofsky's contributions to exemplar-based modeling are highly relevant to the exemplar-based process models described in Section 6.4 First, the GCM relaxes the Context Model's restriction to Boolean-valued attribute dimensions; the GCM's reformulation of the multiplicative similarity rule allows it to be applied to domains with numeric-valued attribute dimensions. Second, Nosofsky adopted the *attention-optimization hypothesis* in the GCM, which states that humans selectively attend to attributes to maximize their categorization performance. Selective attention, which leads to systematic changes in the structure of the psychological space (Shepard, 1964), is achieved by modifying the GCM's attribute weight settings. These weights are not learned, but are instead computed by a *multi-dimensional scaling* (MDS) procedure that determines the weight settings from subjects' similarity assessments for all pairs of instances in the instance space (Carroll & Wish, 1974). Although the attention-optimization hypothesis was explored previously by Reed (1972), his exemplar-based models did not incorporate Shepard's law.

Nosofsky's model depends on the settings for a large number of parameters (i.e., the scaling parameter and the attribute weight parameters). This property of categorization models is generally regarded as problematic; given an infinite number of parameters, a model can fit any subject data. Furthermore, although Nosofsky posited that humans employ a selective attention process for setting these parameters, he offered no process algorithm for setting these weights. Finally, although Nosofsky noted that similarity is context-dependent (Tversky, 1977; Barsalou, 1982; Roth & Shoben, 1983; Medin & Edelson, 1988), Aha and McNulty (1989) pointed out that the GCM requires a unique set of attribute weights for each target concept because the weights, which reflect an attribute's classificatory significance, do not necessarily have the same setting for each target concept. Aha and Goldstone (1990)

[3] I am ignoring Nosofsky's (1986) concept bias parameters here, which modify the prediction function based on the pre-determined bias for predicting that a member is in a given category. By ignoring them, I assume that they have equal value.

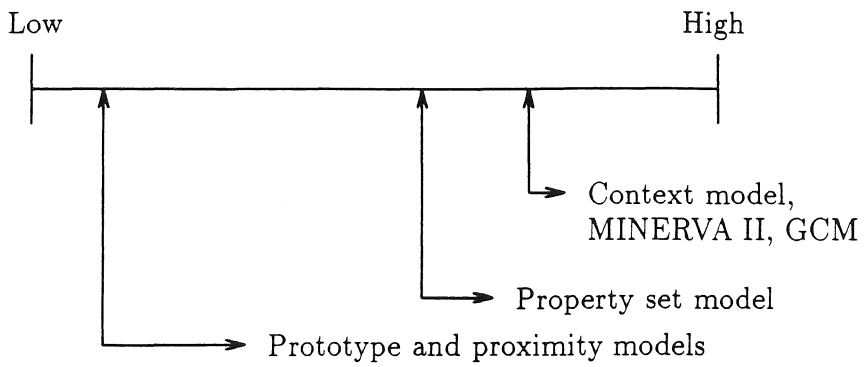


Figure 6.3: Exemplar-based models' relative utilization of attribute correlation information.

further argued that a separate set of weights is needed for each instance to capture the notion that an attribute's classification relevance varies according to the settings of an instance's other attribute values. These extensions require the introduction of a tremendous number of parameters and shed doubt on the GCM model's capability to extend to more complex categorization tasks. However, Section 6.4 describes experimental evidence that process models for extensions of the GCM with category-specific and instance-specific attribute weighting strategies significantly improve its capability to fit subject data.

Summary

Figure 6.3 summarizes the relative use of correlational information of the three exemplar-based models described in this section with three previous models of categorization. The context model, MINERVA II, and the generalized context model all employ Shepard's (1987) law and they all support processes for selectively attending to attributes. However, MINERVA II supports selective attention by spreading activation between similar instances whereas the other two models modify explicit attribute weights. Therefore, they appear to use correlational information equally well. All three models utilize correlational information better than the property set model, which does not support selective attention.

6.3.4 Connectionist Models

Connectionist models date at least to Rosenblatt's (1962) description of the perceptron convergence theorem. A perceptron is a linear separating function that yields a value based

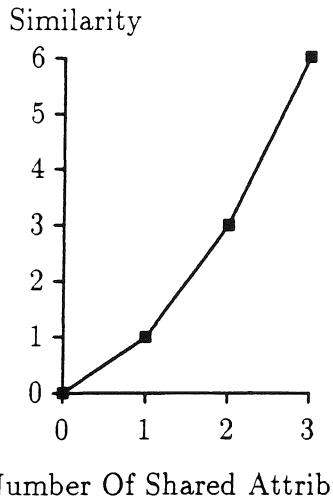


Figure 6.4: The configural cue model defines similarity to increase exponentially with the number of shared attribute values. This graph describes the similarity of any instance with all other instances in the 3-dimensional space in terms of their number of shared primitive and pairwise conjunctive attributes.

on a weighted sum of its input values. A perceptron can be trained to learn weights corresponding to a hyperplane in the instance space, which can be used to partition instances into two categories. Perceptrons utilize correlational information in a manner similar to Reed's (1972) weighted prototype model. Both models maintain a probabilistic view and are restricted to learning linearly separable categories.

Gluck and his colleagues (Gluck, Bower, & Hee, 1989) examined the behavior of a perceptron that inputs values for both primitive attributes and for their pairwise conjunctions. This model, which was named the *configural cue network*, is in accordance with Shepard's law because the number of conjunctions and primitive attributes shared between two instances decreases exponentially with their psychological distance. As an example, consider an instance described by three binary-valued attributes. Figure 6.4 describes the similarity of one such instance with all others that are encodable using three binary-valued attributes. As shown, its similarity with other instances increases exponentially with their number of shared attribute values. In this regard, the configural cue network represents an extension of the perceptron that is similar to Neumann's (1974) model, which extended the simple feature frequency models by adding counts for pairs of features. Like Neumann's model, the configural cue network greatly increases the utilization of correlational information. The configural cue model qualitatively simulated learning curves collected from subjects in Medin and Schwanenflugel's (1981) experiments, where subjects often learned non-linearly separable categories faster than linearly separable categories. It also recorded significantly higher

rank-order correlations than the single cue model with subject data from Hayes-Roth and Hayes-Roth (1977) experiments. Finally, it provided better fits to subject data in an experiment that was used to show that people are sensitive to attribute correlations (Medin *et al.*, 1982).

Kruschke (1990) recently introduced ALCOVE, a connectionist model of category learning that was directly inspired by Nosofsky's (1986) work on the GCM. ALCOVE is a variant of the standard back propagation algorithm (Rumelhart *et al.*, 1987) that uses radial basis functions rather than perceptrons for its hidden nodes. Radial basis functions compute a weighted distance function of its input values to a pre-specified point in the instance space. (The general idea is to associate a point with each radial basis hidden node in such a way that a "good" covering of the space is obtained – one that will support accurate classification behavior.) Therefore, these functions are similar to exponentially decreasing similarity functions such as the one used by Nosofsky (1986) in the GCM. ALCOVE also learns attribute relevance weights on its input attributes. Kruschke (1990) evaluated ALCOVE's ability to display a wide range of psychological phenomena. It simulates all of the behavioral phenomena exhibited by configural cue networks, simulates base-rate neglect exhibited by subjects in Gluck and Bower's (1988) studies, captures the entire range of typicality effects, and, unlike configural cue models, can qualitatively simulate the asymmetric similarity ratings recorded in Tversky's (1977) experiments.

Kruschke found that ALCOVE's performance is qualitatively indistinguishable from a variant of his model that used what are essentially specific exemplars for each of its hidden nodes. In fact, the exemplar-based hidden nodes slightly improved qualitative fits in the experiments on which both variants were tested. It also accounted for some base-rate effects that Gluck and Bower (1988) claim were unaccountable by exemplar-based models. However, it can not account for the inverse base-rate effects exhibited by subjects in an experiment run by Medin and Edelson (1988). They found that an extension of the context model can account for these effects if instance-specific weights are used, which allow an attribute's relevance to vary depending on the values of other attributes. It appears that ALCOVE, in its current form, does not utilize instance-specific correlational information. Kruschke (personal communication) notes that ALCOVE can be extended with what are essentially instance-specific weights and plans to explore this issue in the future.

The relative amount of correlational information utilized by the three connectionist models mentioned in this section is shown in Figure 6.5. As in each previous case, models that utilize more of the attribute correlation information in the training data can explain larger sets of psychological phenomena. For example, the configural cue model extended the perceptron model's representation by adding higher-order inputs, which greatly increased the correlational information that is utilized by the algorithm and allowed it to simulate phenomena not explainable by the perceptron model. ALCOVE, which is closely related to Nosofsky's (1986) GCM model, extends the configural cue model by adding layers of radial basis hidden nodes. This allowed it to explain phenomena not explainable by configural cue

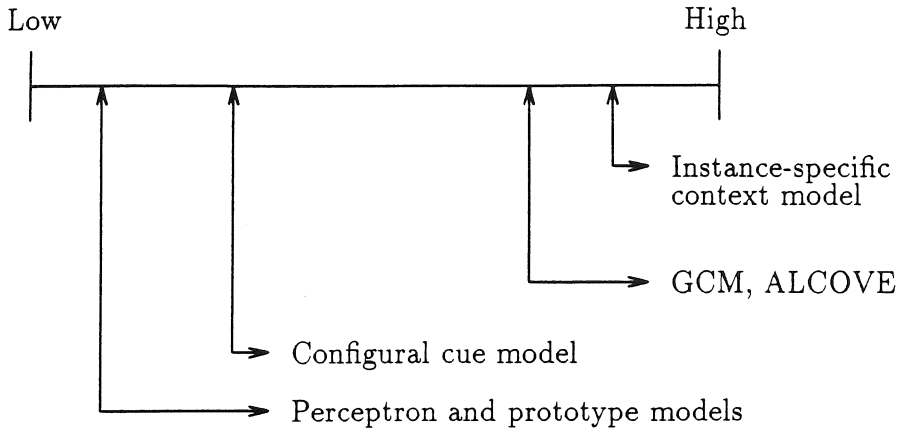


Figure 6.5: Connectionist models' relative utilization of attribute correlation information.

networks (e.g., base rate neglect). However, in its current form, it doesn't appear to be able to use instance-specific correlational information in the extended context model described by Medin and Edelson (1988). Nonetheless, Kruschke's process model represents an exciting advancement in understanding how connectionist networks can model a wide range of psychologically plausible behavior. It is probable that this relatively new area of study will lead to models that can explain previously unaccountable behavior in the near future.

6.3.5 Summary

In summary, research on psychologically plausible models of categorization has progressively relaxed the definition of concepts. The classical view, which states that a concept is defined by a set of necessary and sufficient features, has been replaced by both the probabilistic view, which relaxes the requirement on features, and the exemplar view, which also relaxes the requirement that a concept description be an abstraction. I summarized evidence indicating that there is a monotonic relationship between the models' capability to explain psychological phenomena and their ability to use correlational information contained in the training instances. Four models have received the majority of attention in the literature (i.e., prototype, feature frequency, exemplar-based, and connectionist models). Prototype models, which embody a probabilistic view, were abandoned because they do not retain specific instance information and are insensitive to the variances in a concept's exemplar information. Feature frequency models distribute a category's representation among a set of feature counts, thus relaxing the notion of a centralized abstraction. Models that save higher-order feature frequency counts store specific instance information, but they have not

yet been extended to model selective attention. Exemplar-based models, such as the context model and generalized context model, provide excellent fits for subject data in a wide variety of experiments, can model selective attention, and are in accordance with Shepard's numerous empirical observations suggesting that similarity decreases exponentially with psychological distance. Although it is probable that people retain and use abstractions to solve categorization tasks, exemplar-based models can fit the behaviors normally attributed only to mixed representation models. Finally, a recently developed connectionist process model that uses exemplar-based nodes can account for a wide variety of psychological phenomena. However, it has not yet been developed to exploit instance-specific correlational information.

In fact, no process model has been developed to account for evidence that attribute relevance varies depending on the context of the classification task. Therefore, existing models cannot be expected to provide accurate fits when attribute relevance varies according to context, which occurs frequently in real-world classification tasks. Section 6.6 describes a process model that extends Nosofsky's (1986) generalized context model so that it can fit subject data when learning requires context-specific strategies for selective attention.

6.4 Exemplar-Based Process Models

Barsalou (1989) noted that at least three key results have been learned during the past two decades of research on models of categorization:

1. people use specific instance information in categorization decisions,
2. people use attribute correlation information to derive categorization decisions, and
3. people represent categories with dynamic rather than static representations.

Current process models of categorization must minimally support these behaviors to be considered psychologically plausible. Barsalou's chapter can be interpreted as a plea to fellow researchers to end their bickering concerning the relative strengths of classes of models and instead focus on models that have the following properties:

1. they inspire empirical and theoretical progress,
2. they have plausible assumptions,
3. they develop the space of category learning models, and
4. they are inspired by motivations outside the cognitive psychology literature, including computational tractability.

This section summarizes the development of a research line that lead to the development of a model with all four of these properties.

Table 6.3: Weighting strategies for the four GCM-based process models.

Name	Attribute Weighting Strategy
GCM-**NW**	**N**o **W**eights
GCM-**SW**	**S**ingle Set of **W**eights
GCM-**MW**	**M**ultiple Sets of **W**eights
GCM-**ISW**	**I**nstance-**S**pecific **W**eights

More specifically, this section introduces a sequence of four exemplar-based process models based on Nosofsky's (1986) GCM and summarizes simulations that distinguish their abilities. All four models linearly normalize numeric-valued attributes and use the context model's prediction function, which is Equation 6.1 on Page 6.1. The four models are distinguished from each other by their attribute weighting strategies and are distinguished from the previously described IBn algorithms primarily in that they save all training instances and use the context model's prediction function.

6.4.1 Summary Descriptions

The four models' weight-learning strategies are summarized in Table 6.3. Each algorithm in the sequence uses an extension of the previous algorithm's attribute weighting strategy. The first model, named *GCM-NW*, uses no weights. More specifically, all of GCM-NW's weights are set to $\frac{1}{|P|}$, where P is the set of predictor attributes, and are not modified during the training process. This model was included to determine determine whether attribute weighting is useful. The second model, named *GCM-SW*, is a process model for Nosofsky's GCM model, except that concept bias parameters are assumed to equal.[4] This model should learn accurate concept descriptions more quickly when categorization relevance varies among the predictor attributes. The third model is named *GCM-MW* (Aha & McNulty, 1989). It uses a separate set of attribute weight settings for each target concept and is expected to increase learning rates when both the prediction task involves learning multiple concepts and the categorization relevance of the attributes varies among target concepts. The fourth and last algorithm is named *GCM-ISW*. It uses a separate set of attribute weights for each ⟨instance,target-concept⟩ pair. This model should learn more quickly than the others when attribute relevance varies among the instances in the application.

[4]Concept bias parameters reflect category frequencies, but not association strength. ALCOVE (Kruschke, 1990) learns such error-driven association strengths for each target category. Since ALCOVE permits these strengths to be either positive *or* negative, it can simulate base-rate neglect behavior. The GCM model does not have this capability. However, exemplar-based models appear to be able to model this phenomenon; Medin and Edelson (1988) described extensions of the context model that uses instance-specific weights to help simulate base-rate neglect.

An example should clarify why these models' learning behaviors should be distinguishable. Consider the problem of predicting whether a pro-life politician will endorse proposed legislation on abortion rights. As with most real-world categorization tasks, some attributes should be given more attention than others. In this case, attributes such as "past voting record" should be weighted more than attributes such as "height." An attribute's relative predictive relevance depends on the prediction task (Aha & McNulty, 1989; Aha, 1989a) (e.g., "past voting record" is far less relevant than "height" when predicting the ability to dunk a basketball). However, an attribute's relevance to a categorization task often also depends on its *context*, which I define as the values of the other attributes in an instance. For example, the relevance of the "past voting record" attribute will be low if the "percentage of pro-choice constituency" attribute has a high value (due to pressure from pro-choice political action groups). However, it will be high if the "seek re-election" attribute value is "false", which diminishes the influence of political action groups. Context sensitive attribute weights are required to derive an appropriate psychological space and satisfy the attention-optimization hypothesis when attribute relevance is context-dependent.

Figure 6.6 should help to clarify the relationships between and evolution of these four process models. The generalized context model (Nosofsky, 1986) is a generalization of Medin and Schaffer's (1978) context model in which its relationship to Shepard's (1987) universal law of stimulus generalization is made explicit. The GCM-SW algorithm is a process model for the GCM. GCM-NW is a straw man version of the GCM-SW whose weights are fixed. The GCM-MW process model (Aha & McNulty, 1989) was influenced by IB4, which also employs concept-dependent attribute weights. The GCM-ISW model (Aha & Goldstone, 1990) is a direct descendant of GCM-MW that conforms with many studies providing evidence for context-sensitive categorization phenomena. These algorithms differ from the IBn algorithms introduced in Chapters 2 and 4 in that they do not use a normalization function, they define similarity to decrease exponentially with distance, they use Equation 6.1 on page 160 for their prediction function, and they save all training instances.

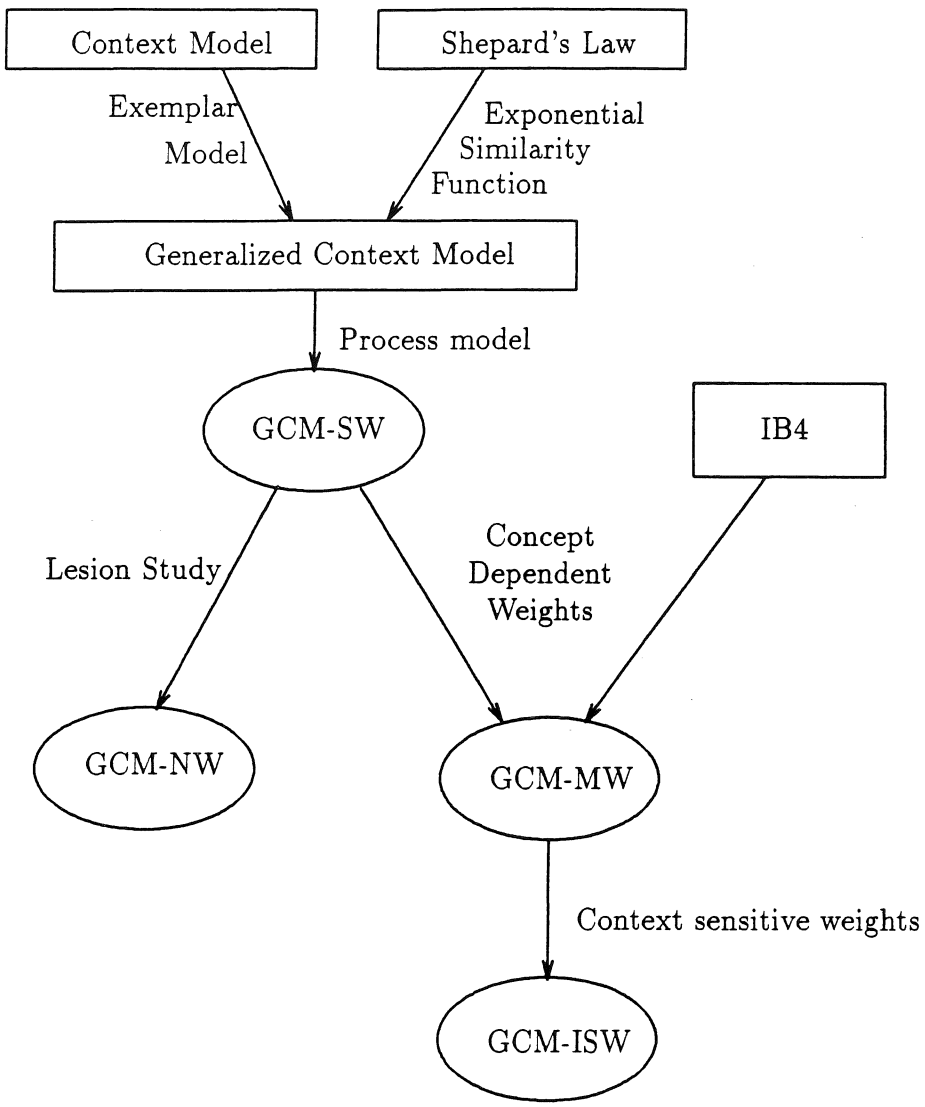


Figure 6.6: Relationships and influences of the four process models.

6.4.2 GCM-SW: Learning Attribute Relevance

GCM-SW's similarity function is

$$\text{Similarity}(x, y, P) = e^{-s\text{Distance}(x,y,P)},$$

where

$$\text{Distance}(x, y, P) = \sqrt{\sum_{i \in P} w_i (x_i - y_i)^2},$$

where parameter s's setting determines the slope of the exponential decay and w_i is GCM-SW's weight for attribute i. Parameter s is set to 10 in all of the simulations and experiments. Values for attribute weights are always initialized to $\frac{1}{|P|}$, range in $[0, 1]$, and are normalized to sum to 1.

GCM-SW's prediction function was taken from the context model. It is

$$\text{Probability}(x, c) = \frac{\sum_{y \in S(c)} \text{Similarity}(x, y)}{\sum_{y \in S} \text{Similarity}(x, y)}, \tag{6.18}$$

where S(c) is the set of stored exemplars in concept c's description and S is the set of all stored exemplars.

GCM-SW's memory updating function modifies the attribute weights for target concept c's partial concept description (PCD_c) based on the the current training instance x's similarities with stored instances in PCD_c. It then linearly normalizes c's attribute weights and adds x to c's description. The GCM-SW training algorithm is:

1. $\text{PCD}_c \leftarrow \emptyset$
2. for each $x \in$ training set do
   2.1 for each $y \in \text{PCD}_c$: compute Similarity(x, y, P)
   2.2 for each $y \in \text{PCD}_c$:
      – for each predictor attribute i: Adjust_weight(i, x, y, c)
   2.3 Linearly_normalize_weights(x, y, c)
   2.4 $\text{PCD}_c \leftarrow \text{PCD}_c \cup \{x\}$

As in IB4, attribute weights denote the estimated relevance of an attribute for a categorization task. However, a different weight-learning algorithm is used in these algorithms; each predictor i's weight is computed using a function of the estimated conditional probability that two instances will have the same class, given that their similarity is high and the difference of their values for i is small. If this probability at time t is denoted by $\text{Pr}_i(t)$, then the attribute weight for i after t training instances is computed by the Adjust_weight algorithm as follows:

1. if $(x_i = y_i)$ then r $\leftarrow$ 1 else r $\leftarrow$ 0
2. $\Pr_i(t+1) \leftarrow \Pr_i(t) + (r - \Pr_i(t)) \times \text{Similarity}(x, y, P) \times e^{-s|x_i - y_i|} \times \rho$
3. $w_i \leftarrow \Pr_i(t) - (1 - \Pr_i(t))$

where ρ is the learning rate parameter, which is set to 0.01 for GCM-SW and GCM-MW in the simulations and experiments in Sections 6.5 and Section 6.6. The size of the update to predictor attribute i's conditional probability increases exponentially with linear decreases in both Distance(x, y, P) and the attribute-value difference $|x_i - y_i|$. Therefore, attribute i's weight is most strongly influenced by highly similar instances with similar values for i.

Instances can either be members or non-members of a concept, corresponding to "positive" and "negative" target values. Instance x is predicted to be a member of a concept c only if the sum of its similarities to c's positive instances is greater than to c's negative instances. Given a target attribute c, the testing algorithm used for all four of these process models is (1) compute the current training instance x's similarity to the instances in c's concept description, (2) compute the probability that x_c is "positive", and (3) output "positive" if this probability is above 0.5 (otherwise, output "negative").

6.4.3 GCM-MW: Learning Concept-Dependent Relevance

The GCM-MW model's concept-dependent similarity function is

$$\text{Similarity}(c, x, y, P) = e^{-s\text{Distance}(c,x,y,P)}, \tag{6.19}$$

where w_{c_i} denotes the weight of attribute i for target concept c in

$$\text{Distance}(c, x, y, P) = \sqrt{\sum_{i \in P} w_{c_i}(x_i - y_i)^2}. \tag{6.20}$$

The GCM-MW model should outperform the GCM-SW model when attribute relevance varies among target concepts. However, GCM-MW's assumption that an attribute's relevance is invariant across all instances is easily violated. For example, attribute relevance can vary among a concept's disjuncts. Furthermore, it can also vary *within* a disjunct. Figure 6.7 displays a two-dimensional domain containing three disjuncts of a single target concept. The horizontal attribute is more relevant than the vertical for disjunct A: small perturbations in the horizontal's values will more frequently change disjunct membership status than will perturbations in the vertical's values. The vertical attribute is more relevant for B while C's attributes are approximately equally relevant. However, attribute relevance differs greatly among instances. For example, although both attributes are relevant for classifications made by instance w, the horizontal attribute is more relevant for x and less relevant for y. Finally, z's vertical attribute is more relevant. GCM-MW's learning rate can be significantly reduced in these situations.

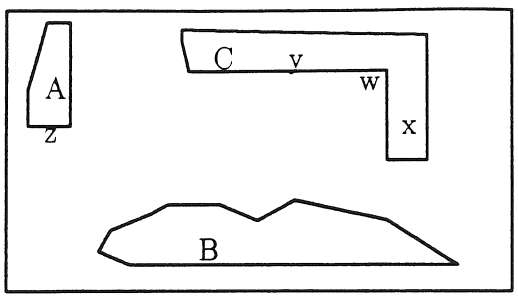


Figure 6.7: Attribute relevance can vary among instances.

6.4.4 GCM-ISW: Learning Attribute Relevance in Context

The GCM-ISW model differs from GCM-MW in that it learns sets of *instance-specific* attribute weights, one set for each ⟨attribute,target⟩ pair. This provides greater flexibility than found in GCM-MW; GCM-ISW relaxes the assumption that attribute relevance is invariant among a target concept's saved instances and uses an instance-specific selective attention mechanism.

GCM-ISW's instance-specific weights can be easily misapplied. For example, consider the set of disjuncts shown in the instance space in Figure 6.7. If GCM-ISW decides that the only relevant attribute for instance z is the vertical attribute, then z will appear to be very similar to x, which is located far from z in this instance space. This implies that instance-specific weights should be used only when the instance being classified is highly similar to the classifying instance. GCM-ISW solves this problem by learning both concept-dependent weights (as is done in GCM-MW) and a separate set of instance-specific weights. GCM-ISW's similarity function then dynamically combines these two sets of weights to compute the *context*-specific similarity of two instances as follows:

$$\text{Distance}(c, x, y, P) = \sqrt{\sum_{i \in P} \text{Combine\_weights}(c, x, y, i) \times (x_i - y_i)^2}.$$

When computing the similarity of a new instance x to previously processed instance y, Combine_weights calculates attribute i's *context-specific* weight as follows:

$$\text{Combine\_weights}(c, x, y, i) = (w_{c_i}(y) \times \text{scale\_factor}) + (w_{c_i} \times (1 - \text{scale\_factor})),$$

where $\text{scale\_factor} = (1 - |x_i - y_i|)^r$, $w_{c_i}(y)$ is i's attribute weight for saved instance y, and r is a combination parameter that determines the relative impact of the concept-dependent and instance-specific attribute weights in calculating the context-sensitive weight. Combine_weights uses instance-specific weights more confidently when the difference of the

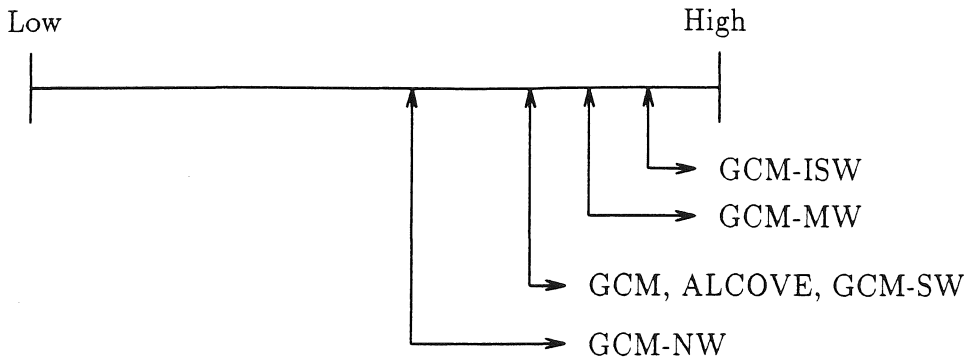


Figure 6.8: The exemplar-based process models' relative utilization of attribute correlation information.

values for i is small. This reduces the frequency with which instance-specific weights are used when the distance between instances is large. After GCM-ISW computes similarities, it updates the conditional probabilities and attribute weights for both its concept-dependent and its instance-specific weights.

The combination parameter r was set to 0.5 in the simulations and experiments described in the following sections. GCM-ISW's learning rate parameter ρ was set to a higher value (i.e., 0.1) than its value in the other models when it updates instance-specific weights (the value 0.01 was still used to update concept-dependent weights). This is needed because, given any one training instance, few other training instances are highly similar to it. However, when updating concept-dependent weights, there will be several highly similar pairs of instances, which allows it to be updated more quickly even though the GCM-SW and GCM-MW models use a lower setting for learning rate.

6.4.5 Summary

Figure 6.8 summarizes the relative use of relational information for the four exemplar-based process models. The GCM-SW model uses the same information as the GCM. GCM-NW uses less since it cannot selectively attend to attributes. GCM-MW is an improvement over GCM-SW since its selective attention mechanism allows it to distribute attention differently depending on the target concept. Finally, GCM-ISW uses the most relational information since it learns this information for each stored instance. This advantage will allow the GCM-ISW model to outperform the other models in the simulations in Section 6.5 and experiments in Section 6.6.

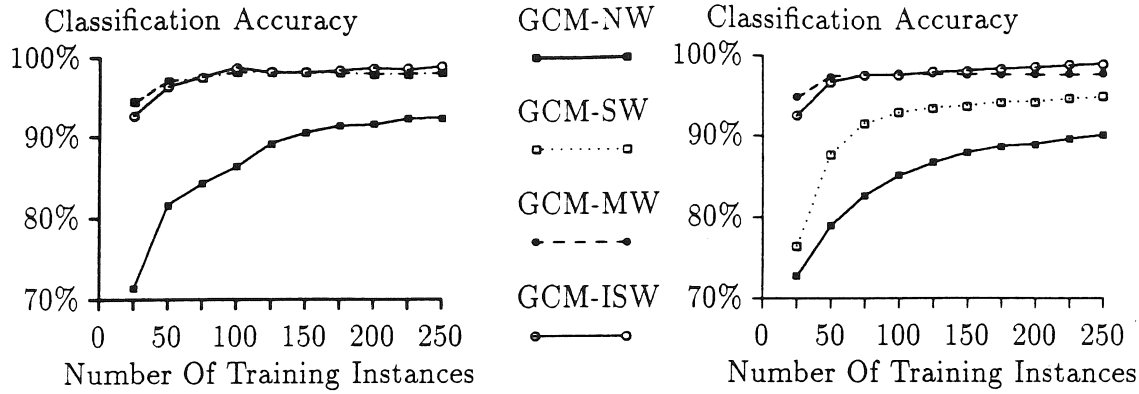


Figure 6.9: Learning curves for the four IBL algorithms. Left: GCM-NW learns slowly when attributes have different relevance. GCM-SW and GCM-MW behave identically in this simulation since there is only one target concept. Right: GCM-SW learns slowly when each attribute's relevance differs among target concepts.

6.5 Simulations

The four exemplar-based process models introduced in the previous section were applied to three artificial domain that were designed to highlight flaws in the designs of GCM-NW, GCM-SW, and GCM-MW respectively. All of the results described in this section were derived from averaging over 20 pairs of training and test sets with 250 and 100 instances respectively. Values for predictor attributes were randomly selected from the range $[0, 1]$ according to a uniform distribution.

The first simulation highlights the utility of using attribute weights. GCM-SW's attribute weights are useful when attribute relevance varies among predictors. It's performance was compared with GCM-NW's, whose weights remain fixed with value $\frac{1}{|P|}$. GCM-NW learns slowly when attribute relevance differs among the predictors. The graph in the left of Figure 6.9 shows the average learning curves for a simulation with one target concept and ten numeric-valued predictors, only one of which was relevant. Target concept members were defined to be those whose relevant attribute's normalized value was greater than 0.5. As expected, GCM-SW's average accuracy (measured across the ten applications to the test set per trial) was significantly greater than GCM-NW's ($t(19) = 4.54, p < 0.001$).

However, since the GCM-SW model uses the same setting of attribute weights for all targets, it performs relatively poorly when the relative relevance of attributes differs greatly among target concepts (Aha & McNulty, 1989) or when relative attribute relevance varies among instances. The right-hand graph in Figure 6.9 shows the average learning curves when the artificial domain was extended to contain an additional three target concepts, where each of the four target concepts have a single (different) relevant predictor. GCM-SW's

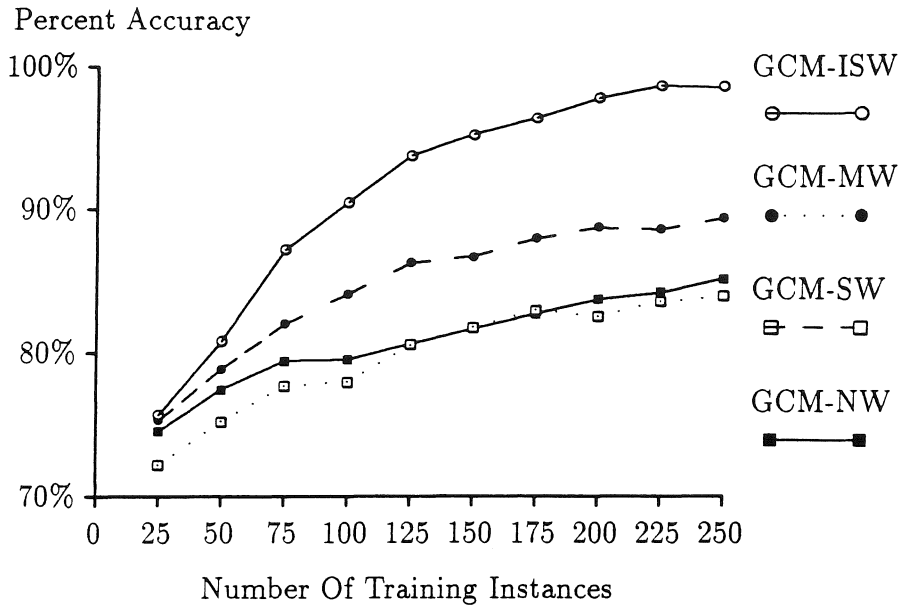


Figure 6.10: Learning curves for the simulation in which attribute relevance varies among instances.

learning curve rises slowly because it is unable to learn an attribute's concept-dependent relevance: its weights for the four relevant attributes each converge to 0.25. GCM-SW's average classification accuracy is significantly lower than GCM-MW's ($t(19) = 5.33, p < 0.001$) in this simulation.

However, the GCM-MW model does not perform particularly well when attribute relevance varies among instances. Figure 6.10 displays the average learning curves when the learning task is changed so that each of the four concepts is defined by a set of five disjuncts. In this case, each disjunct is defined by a single relevant attribute and each attribute in the domain is relevant to exactly two disjuncts overall. The threshold values for inclusion in a disjunct were below 0.14 for the disjuncts of the first two target concepts and above 0.86 for the latter two target concepts. GCM-MW's average accuracy is significantly lower than GCM-ISW's ($t(19) = 3.85, p < 0.002$) in this simulation.

In summary, the GCM-ISW model performed significantly better than the other models in the third simulation and performed as well as the GCM-SW and GCM-MW models in the first and second simulations respectively. However, this does not necessarily mean that it will fit subject data better. The following section describes two experiments with these models. The experimental results provides evidence the GCM-ISW model yields significantly better fits than the other models to subject data in applications where attribute relevance varies among instances.

6.6 Experiments

The GCM-based process models described in the previous section were evaluated against subject data collected from two experiments designed to encourage the subjects to assign different relevance to an attribute depending on its context (i.e., the instance's other attribute values). The GCM-ISW model was expected to provide the best fits to the subject data.

6.6.1 Experimental Design

Forty subjects participated in two experiments (i.e., twenty per experiment). The subjects were students enrolled in an undergraduate course on introductory psychology at the University of Michigan. They were compensated with course credit for participating in the study.

Both experiments used the same instance space. The space has two integer-valued dimensions, each with a range of 8 values. Subjects were trained on 12 training instances in a binary classification task; the choice of training instances distinguished the two experiments. Practice continued until four perfect runs through the set of 12 training examples was completed. These four runs did not have to be contiguous. Subjects were tested once on each instance in the space. Both training and test items were shown in random order, including a shuffling of the training items for each presentation cycle. The subjects were trained and tested and their classification responses were gathered using a computer program developed by Robert L. Goldstone of the University of Michigan. The two dimensions were line position, which move from left to right across the field of view, and the size of a square shown on the video screen. Scales were shown next to the training and test instances so that subjects could more easily estimate square height and line position.

The models were trained and tested in the same manner except that their instances are represented as vectors in a two-dimensional space and they were directed to yield the *probability* that they would classify an instance as a member of category 2 rather than a binary classification guess. GCM-MW will be indistinguishable from GCM-SW in this task since they only differ when the prediction task involves more than two target concepts. Therefore, fits will be described for only three of the models. Furthermore, the GCM-NW model is expected to perform similarly to the GCM-SW model because the two predictor attributes appear to be equally useful for classification purposes in applications where attributes have, on average, relatively equal relevance for predicting accurate target values. Since GCM-SW's attribute weights are expected to be relatively equal for the two attributes, its should behave similarity to the GCM-NW model.

The training sets for the two experiments are shown in Figure 6.6.1. The twelve training instances in each experiment are labeled with their category name (i.e., "1" or "2"). Question

Experiment #1:

	1	2	3	4	5	6	7	8
1	.	.	.	.	.	.	.	.
2	.	.	.	.	.	2	?	.
3	.	.	.	.	.	2	1	.
4	.	.	.	.	.	2	1	.
5	.	.	.	.	.	?	1	.
6	.	1	1	1	?	.	.	.
7	.	?	2	2	2	.	.	.
8	.	.	.	.	.	.	.	.

Experiment #2:

	1	2	3	4	5	6	7	8
1	.	.	.	.	.	.	.	.
2	.	.	.	2	2	2	?	.
3	.	.	.	?	1	1	1	.
4	.	1	?	.	.	.	.	.
5	.	1	2	.	.	.	.	.
6	.	1	2	.	.	.	.	.
7	.	?	2	.	.	.	.	.
8	.	.	.	.	.	.	.	.

Figure 6.11: Training sets and critical test instances for the two experiments.

marks indicate test instances of key interest. In the first experiment, the vertical axis appears to be the only relevant attribute in the lower left portion of the space while the situation is reversed in the upper right portion of the space. That is, subjects are expected to state that the test instance in row seven, column two ([7, 2]) is a member of category 2. Similarly, subjects are expected to state that the test instance located at [5, 6] is also member of category 2. However, both [6, 5] and [2, 7] are expected to be labeled with category 1. In the second experiment, the expected responses for test instances [3, 4] and [7, 2] is category number 1, while the expected responses for [2, 7] and [4, 3] is category number 2.

6.6.2 Results

Fisher's method for converting correlations (r) to Z-scores was used to evaluate the fits of each model to the subject data. The detailed results are listed in Appendix B. The correlation between GCM-ISW's results and the averaged subject data for the 64 test instances was 0.81 and 0.85 for the first and second experiments respectively. GCM-MW's was 0.66 for both of the experiments and GCM-NW's was 0.65 and 0.68. The GCM-ISW model's results correlated significantly better with the subject data from the first experiment than did the GCM-NW ($Z = 2.75, p < 0.01$) and the GCM-MW ($Z = 2.61, p < 0.01$) models. GCM-ISW's results for the second experiment also had significantly better correlations with the subject data than did the GCM-MW and GCM-NW algorithms (i.e., ($Z = 3.62, p < 0.0005$) and ($Z = 3.36, p < 0.002$) respectively).

Furthermore, GCM-ISW performed even better for the four highlighted test instances. Table 6.4 summarizes the results for these instances in both experiments. Although it is unusual to compute correlations over only four instances, the following results highlight the fact that GCM-ISW provided significantly better fits to the subject data than did the other models. Its correlations with the average of the subjects' predictions for these two sets of four test instances was 0.97 and 0.95 respectively. GCM-MW's respective correlations were

Table 6.4: Model predictions and percent subjects voting for category number 2 for the four critical test instances in each experiment.

Experiment	Location	GCM-NW	GCM-MW	GCM-ISW	Subjects
1	[2, 7]	62	66	33	20
	[5, 6]	40	35	57	70
	[6, 5]	47	42	37	10
	[7, 2]	50	55	75	90
2	[2, 7]	53	47	76	90
	[3, 4]	44	51	36	40
	[4, 3]	37	43	58	50
	[7, 2]	64	59	31	15

0.17 and -0.84 while GCM-NW's were -0.42 for both experiments. Although this is a small number of instances, GCM-ISW's correlations were significantly better than the GCM-MW and GCM-NW's models for both the first ($Z(1) = 1.92, p < 0.1; Z(1) = 2.53, p < 0.025$) and second ($Z(1) = 3.05, p < 0.0025; Z(1) = 2.28, p < 0.025$) experiments respectively. Both the majority of subjects and the GCM-ISW model guessed these eight test instances would have the values described earlier. The GCM-MW model agreed on only two of these eight instances while the GCM-NW model agreed on only four.

These results provide evidence that the GCM-ISW model is a more psychologically plausible model than its predecessor models. The combination of concept-dependent and instance-specific attribute weights captures the context sensitivity of attribute relevance in these experiments. In summary, these results support the claim that *a psychologically plausible learning algorithm's selective attention processes must be a context-dependent function*; a simple strategy of using one weight per attribute will not necessarily provide optimal fits to subject data.

6.7 Discussion

6.7.1 Evidence of Context-Dependent Categorization

Tversky (1977) was one of the first advocates of the notion that the salience of an attribute is determined in part by the context of the categorization task. Tversky's definition of context concerns the situation in which a question is asked. This can be related to my definition for context as follows. Exemplar models interpret questions are referring to specific target attributes. In the GCM-ISW, these attributes are associated with a set of attribute weights for each instance. Using this interpretation, a question-specific context directly relates to GCM-ISW's definition of context. Tversky described evidence that

a two-way relationship exists between similarity and classification: pairwise similarities are modified by experience with classification attempts. This provides support for the plausibility of the latter three GCM-based models, which modify similarity parameters in response to classification attempts. However, Tversky's main result was that his *contrast model* of similarity accounts for empirical evidence that subjects' similarity functions are *asymmetric* (i.e., Similarity$(x, y) \neq$ Similarity(y, x)). The contrast model can account for asymmetries in subjects similarity ratings because it defines similarity as a function of two instances' shared features minus the ones they do not share. Furthermore, three parameters are used to determine the relative weight of the shared features, the features held only by the first instance, and the features held only by the second instance. Asymmetries result by varying these parameters' settings depending on the ordering of the two instances in the similarity computation. The GCM-ISW models' similarity function is also asymmetric because it is instance-specific. That is, it uses a different set of instance-specific weights for each instance. Asymmetries result whenever these weights are different for the pair of instances being compared. More experimentation would be required to determine whether GCM-ISW's similarity function provides good fits for subjects' similarity ratings, but it certainly is an improvement over the similarity functions used in the context model and the GCM, both of which cannot account for these asymmetries.

Many other researchers agree that models of categorization should be context-sensitive. Barsalou (1982) argued that an instance's context influences its perceived typicality and determines which of its attributes receives attention. For example, he noted that, while some attributes of "basketball" (e.g., "round") are always salient, others (e.g., "floats") only become salient (i.e., quickly retrieved) in contexts involving water. This provides psychological support for the GCM-ISW model: people on a luxury liner attend more to the "floats" attribute when the ship is sinking (to judge whether objects are members of the "can support me in the water" category) than when it is in port. Barsalou also argued that concepts contain both context-dependent and context-independent properties and that only the latter is activated independent of context.[5] Roth and Shoben (1983) and Caplan and Barr (1988) described similar results in which context affects typicality ratings. Furthermore, Roth and Shoben argued that typicality has no effect on the time required to identify a concept exemplar independent of context while Caplan and Barr argued that context does not affect category structure (i.e., its intrinsic features, which are true of the category in isolation), but only its extension. Ortony, Vondruska, Foss, and Jones (1985) argued persuasively that models of similarity must assume that the relevance of an attribute can vary across instances and that the relevance of an attribute for a given instance can vary across contexts. Finally, Goldstone, Medin, and Gentner (in press) argue that, when comparing instances, the influence that one attribute has depends on the other attributes that are shared by the instances. In summary, concepts are no longer regarded as static definitions, but rather as dynamic, context-dependent representations of categories (Barsalou & Medin, 1986).

[5]This notion is similar to Michalski's (1990) arguments for a two-tiered concept representation, in which a concept has an identifiable core and a set of procedures for usage, one per context.

6.7.2 Other Context-Specific Exemplar-Based Models

Three other exemplar-based models that relax the restriction of one attentional weight per attribute dimension have recently been described in the literature. The common thesis of these models is that selective attention processes can mediate the tension between developing general abstractions and retaining specific instances. First, Nosofsky, Clark, and Shin (1989) considered *value-specific* weighting algorithms in which each attribute value would be associated with a unique weight setting. This strategy is not as flexible as instance-specific weighting algorithms: value-specific weights for some attribute i will not work well when i's relevance varies over instances that have the same value for i. A better suggestion is Medin and Edelson's (1988) proposal to use context-sensitive retrieval processes in the process model. Their model, which is the only model found to account for subjects' context-specific sensitivity to base rate information during categorization tasks, is extremely similar to the GCM-ISW model in that it assumes that similarity parameters are associated with specific exemplar representations rather than with attribute dimensions in their entirety. However, their proposed process model differs from GCM-ISW's learning algorithm. When an instance is correctly classified, their model assigns high relative weights to the attributes shared by the classifying instance and the instance being classified. Misclassifications result in assigning higher weights to attributes that are *not* shared by these two instances. This is in contrast with the GCM-ISW model, which also reduces the weights for attributes that (1) are not shared during correct classifications and (2) that are shared during misclassifications. It is not obvious what the comparative benefits are of these two sets of assumptions, but experience with the GCM-ISW model suggests that their proposed model's similarity function should be extended to ensure that instance-specific weights are not used to classify instances located in distant regions of the instance space. Finally, Medin and Shoben (1988) present examples that suggest an *instance-directed* attribute-weighting scheme is a promising model that requires further investigation. They found that, while *White* is more similar to *Gray* than is *Black* for the attribute *hair*, exactly the opposite pattern emerges with the attribute *clouds*. That is, gray hair is generally considered to be more similar to white hair than it is to black hair, but gray clouds are generally considered to be more similar to black clouds than to white clouds. This suggests extending the instance-specific weighting method to distinguish between directions along numeric-valued attribute dimensions. For instances of hair, the gray-black distance is widened while the gray-white distance is reduced. In any case, a single predefined weight for the *color* dimension will not survive changes of context.

Models that learn context-specific attribute weights resemble rule-based learning algorithms (Nosofsky, Clark, & Shin, 1989; Aha & Goldstone, 1990). By weighting dimensions selectively on the basis of their category diagnosticity, the exemplar-based systems are qualitatively distinguished from the simple storage of instances in a "raw form." Although instance information is not discarded, it is selectively emphasized. This representation is similar to that used for rules. For example, consider the concept of legal-sized suitcases (i.e., those with lengths less than five feet). An instance-directed weighting algorithm could learn

a high weight for 4′ 9″ in the positive direction and a low weight in the negative direction for legal-sized suitcases. This is similar to the rule "if 4′ 9″ or less, then legal-sized luggage, otherwise illegal."

The GCM-ISW model adds an enormous number of parameters into the GCM model. Although it increases learning rate, its additional parameters are not needed when attribute relevance remains constant across the entire dimension. A more elaborate model would learn which parameters should be permanently fixed without need for subsequent attention. The algorithm would initially assume that all dimensions are weighted equally for all categories. If this assumption does not yield sufficiently fast learning rates, then the system would relax its assumptions and allow an attribute's weight to vary across categories. The assumption that weights are fixed across instances could also be automatically relaxed. Shifts in the target concept description could lead to more or less specific weighting algorithms in attempts to maximize classification accuracy while minimizing the number of unique weights that are postulated.

6.7.3 Other Benefits and Limitations

Psychologically plausible exemplar-based models are not limited to simple categorization tasks. In fact, this approach may be applicable to explaining several other behaviors. For example, one of the main advantages of an exemplar-based approach is that little (if any) specific information is lost. Therefore, these models may explain how people dynamically construct ad hoc categories to achieve their goals (Barsalou, 1983; Kahneman & Miller, 1986). Barsalou argued that people often develop ad hoc categories, which he noted possess graded structure. An example of an ad-hoc category is *things sold at a garage sale.* These categories are assumed to not be represented by abstractions. Instead, they are postulated to be constructed by a dynamic process at the time of categorization. Since exemplar-based process models can describe graded category structures (Aha, 1989a), represent concepts extensionally, and determine category boundaries dynamically, they should prove useful in modeling ad hoc categories. Next, Logan (1989) showed that an exemplar-based approach can simulate the well-known power-function speedup of practice. Furthermore, his algorithm also provides tight quantitative fits to subject data exhibiting this speed up. Finally, Fried and Holyoak (1984) argued that an exemplar-based approach is perhaps the only type of model that can fit subject data in which subjects learn categories that are not normally distributed. They made this suggestion after explaining why their Bayesian model cannot describe this behavior.

However, exemplar-based models are not without their limitations. One limitation is that they do not appear to be good simulators for continuous stimulus-response relations (i.e., numeric prediction tasks). Koh and Meyer (1988) described evidence that an adaptive regression model fits subject data significantly better than a simple exemplar-based model for

three different stimulus-response relations. Although the exemplar-based approach performs well during interpolation, it performs poorly during extrapolation. However, Koh and Meyer did not attempt to modify the simple exemplar-based approach to determine whether a more elaborate variant would provide better fits.

Perhaps the main limitation of exemplar-based models is that they do no not incorporate causal knowledge nor ascribe a role for theories in organizing concepts (Medin & Shoben, 1988). Murphy and Medin (1985) argued that the notion of similarity relationships does not sufficiently constrain which concepts will be coherent and which are meaningful. The core of the problem is that they cannot represent intra- and inter-concept relations and domain specific knowledge in general. This might be debated by Fisher (1989). He argued that his COBWEB system, which learns a probabilistic concept hierarchy whose leaves are specific instances, is an instance-based learning algorithm with an efficient retrieval mechanism.[6] It might also be argued by Bareiss (1989a), who showed how the Protos exemplar-based knowledge-acquisition system can become proficient in the domain of clinical audiology. However, these algorithms bear little resemblance to the exemplar-based models in the psychological literature that have been empirically evaluated for their ability to fit subject data.

6.8 Chapter Summary

This chapter summarized the status of psychologically plausible instance-based learning algorithms, which are normally referred to as exemplar-based process models in the literature on psychological theories of categorization. I also presented a process model for the GCM (Nosofsky, 1986), a popular exemplar-based model, and extended it to account for situations in which attribute relevance varies across target concepts and where attribute relevance is context-dependent (i.e., dependent on the instance's other attribute value settings). While formal psychological models involving context-specific weight learning do not exist, there is a plethora of psychological data suggesting the existence of such specific weighting systems. Results from simulations suggest that previous exemplar models that selectively weight attribute dimensions, while better than no selective weighting at all, can be improved by representing context-sensitive attribute weights. The GCM-ISW model recorded significantly faster learning rates than the other models in the simulations and provided the best fits to the subject data in an experiment where attribute relevance appeared to be context sensitive.

[6]Fisher (1988) also showed how a version of COBWEB can simulate basic level effects (Mervis & Rosch, 1981), where subjects' classify instances as members for some concepts more quickly than for their subordinate or superordinate concepts. COBWEB is the only machine learning algorithm that can currently account for these effects.

Chapter Acknowledgements

Thanks to Dennis Kibler, Pat Langley, Mike Pazzani, and Dale McNulty for inspiring my interest in cognitive science. Dennis, Pat, and Mike have held several seminars during my years at U.C.I. that focused on this topic, and Mike continues to be an informed contributor and source of information on this literature. Dale spent many hours discussing the results of psychological research on related topics. He was also extremely helpful in the development of GCM-MW (Aha & McNulty, 1989). Dale's solid grasp on this literature and his excellent proofreading and drafting skills helped to greatly improve the content and readability of our paper.

The experiments described in this chapter were designed and carried out by my colleague Robert L. Goldstone of the University of Michigan's Department of Psychology. Rob's contribution to this chapter was enormous in that he showed me how to interpret the experimental results and patiently answered hundreds of my questions on cognitive psychology. Thanks also to Steve Hampson, who acted as an informed sounding board during my repeated attempts to bring organization and structure to this chapter. Both Mike and Dennis added suggestions on how to evaluate and present the experimental results.

After having run the simulations, I was delighted to learn that Medin and Edelson had earlier shown that an instance-specific exemplar-based model can account for behavior unaccountable by any other model. The relevant literature on categorization appears to support the properties of the GCM-ISW model. I expect that future models of categorization (e.g., extensions of ALCOVE (Kruschke, 1990)) will strive to incorporate these properties.

Chapter 7
Survey of Related Work

...we might say that the machine is "learning." The machine learns to characterize classes by the selection of "typical" samples from a larger collection of samples that are introduced sequentially.
– George S. Sebestyen (1962, page 103)

Instance-based learning algorithms evolved from influences in pattern recognition, machine learning, and cognitive science. Section 6.3 reviewed psychologically plausible models of categorization based on the storage of specific instances. This chapter summarizes the relationship of the algorithms described in this dissertation to similar algorithms in the pattern recognition and the machine learning literature.

Instance-based learning algorithms are descendants of *edited nearest neighbor* algorithms in the pattern recognition literature. Section 7.1 briefly summarizes the evolution of these algorithms and discusses how their performance criteria differs. Section 7.2 summarizes the contributions of IBL algorithms in the machine learning literature to the framework described in Chapter 2. Finally, Section 7.3 summarizes examples of algorithms in other learning paradigms that depend on the storage of specific instance information to achieve their goals.

7.1 Edited Nearest Neighbor Algorithms

A vast amount of attention has been given to analyzing the behavior of k-nearest neighbor (k-NN) algorithms in the pattern recognition literature. Surprisingly, the development of IBL algorithms in the machine learning literature has proceeded independently of this body of knowledge. This occurred because the goals of the researchers in the two literatures differ considerably. Machine learning researchers are concerned with issues such as overfitting effects, incremental learning, tolerating missing values, and processing symbolic attributes. Researchers in pattern recognition address issues such as ensuring that concept descriptions perfectly classify training data and focus on creating algorithms with decreased upper bounds on misclassification rates, assuming infinitely-sized training sets. Nonetheless,

these two areas have several common concerns (i.e., maximizing predictive accuracy, minimizing resource requirements) and methodologies (i.e., mathematical analyses, empirical validations with both artificial and real-world databases). Furthermore, many pattern recognition researchers have mathematically and empirically analyzed storage reduction strategies for instance-based pattern classification algorithms. This section briefly summarizes their progress.

7.1.1 The Nearest Neighbor Algorithm

As mentioned previously in Section 3.1, Fix and Hodges (1951; 1952) were the first to publish reports on algorithms resembling k-NN. Cover and Hart (1967) were the first to publish detailed analyses of these algorithms. Their most important contribution was showing that the nearest neighbor (NN) algorithm's error rate is within twice that of the Bayes optimal error rate. This rate is the lowest rate of misclassification given complete information concerning the probability density functions for all of the instances in the instance space. Cover and Hart proved that

$$P_B \leq P_{NN} \leq 2P_B \tag{7.1}$$

where P_B is the Bayes optimal error rate. This rate is determined as follows. Given a set C classes, let $P(c)$ be the prior probability of class c, where $c \in C$. Bayes theorem tells us that the conditional probability that an instance x is a member of class c is

$$P(x \in c) = \frac{P(c)PDensity(x, c)}{\sum_{c' \in C} P(c')PDensity(x, c')}, \tag{7.2}$$

where $PDensity(x, c)$ is the probability density of class c at instance x. Let $L(c, c')$ be the *loss* incurred when an instance in class c is incorrectly classified as a member of class c'. In their analysis, the metric loss function L must be transitive, symmetric, have a positive range, and yield 0 only when $c = c'$. If instance x is predicted to be a member of category c', then its conditional loss is

$$Conditional\_loss(x, c') = \sum_{c \in C} P(x \in c)Loss(c, c') \tag{7.3}$$

The Bayes decision rule simply minimizes this value. That is, the Bayes rule's probability of error P_B for instance x is

$$P_B(x) = \min_c \{Conditional\_loss(x, c')\} \tag{7.4}$$

and the overall expected Bayes error rate is

$$P_B = E[P_B(x)], \tag{7.5}$$

where the expectation is computed with respect to

$$\text{PDensity}(x) = \sum_{c \in C} \text{P}(x, c)\text{PDensity}(x, c) \tag{7.6}$$

The value $\text{E}[\text{P}_B(x)]$ can be thought of as a weighted mean of the conditional probabilities that instance x is a member of category c, where the weights are the probability density functions $\text{PDensity}(x, c)$.

Cover (1968) later extended these results when he showed that the nearest neighbor's error rate is twice that of the Bayes optimal rate and the k-NN algorithm's error rate is

$$(1 + \frac{1}{k})\text{P}_B \tag{7.7}$$

Thus, the difference between k-NN's error rate and the Bayes optimal rate decreases exponentially with linear increases in k. The experimental results in Section 5.2.2 reflect this fact; as k was increased, IB1's predictive accuracy initially increased greatly and tended to taper off with higher settings of k. However, there were many cases in which increasing k after some point resulted in *lower* predictive accuracies. This occurred because the domain characterization assumptions used in Cover and Hart's formal analysis are gross simplifications of real-world domains. They assumed that the data was noise-free, that all attributes are relevant, and that the training set is infinite in size. These assumptions do not normally hold for training sets used in practical applications. Nonetheless, their results are valuable because they imply that the k-NN algorithm can often improve on the nearest neighbor algorithm's error rate, which is now known to be bounded above by twice the Bayes optimal error rate.

7.1.2 Storing Only Misclassified Instances

IB2 (Kibler & Aha, 1987) was introduced without any awareness of similar algorithms in the pattern recognition literature. In fact, Hart's (1968) *condensed nearest neighbor* (CNN) algorithm is an iterative variant of IB2 that uses the nearest neighbor function to predict symbolic target values.[1] Although both CNN and IB2 are "failure-driven" (i.e., they store only misclassified instances), CNN differs in that it repeatedly cycles through the data set until none of the remaining training instances are stored during a cycle. Hart found that, like IB2, the vast majority of the stored instances lie near a concept boundary. Hart tested CNN in a letter recognition task, where the letters were selected from nine different font styles and were represented with 96-dimensional binary vectors. In a single run of the program, CNN cycled four times, stored 197 of the 6295 training samples, and recorded a 98.72% classification accuracy on the disjoint set of 5705 test instances. Although this appears to be

[1] IB2 and CNN also differ in that IB2 normalizes numeric attribute values and has defined procedures for processing both symbolic and missing attribute values.

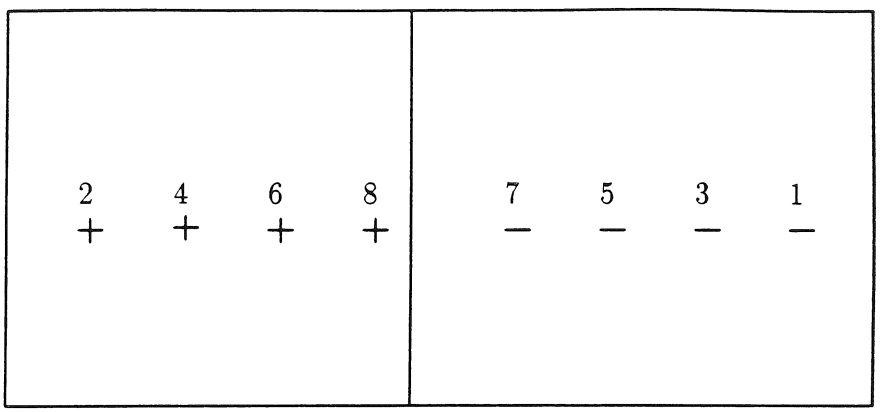


Figure 7.1: Eight instances, labeled in order of decreasing distance from the concept boundary.

a good result, Hart explained that it is "disappointing" since other classification algorithms had recorded superior classification accuracies with this dataset (i.e., in the range 99.5%-99.8%). However, he was surprised and pleased by CNN's low storage storage requirements (i.e., 3.1%).

Hart (1968) lamented the lack of theoretical analysis for CNN and posed four questions for others to consider in its future research:

1. What is its expected storage requirements?
2. How would it perform when $k > 1$?
3. What is its expected increase in error rate in comparison to the nearest neighbor algorithm for a finite-sized training set?
4. How many iterations does CNN require?

The analyses in this dissertation have answered the first two questions. That is, the mathematical analyses in Chapter 3 suggest that CNN's expected storage requirements are polynomial in the size of the target concept's boundary. The empirical analyses in Section 5.2.2 showed that IB2's predictive accuracy increased with increasing values for k and eventually peak before dropping with higher settings for k. However, the experiments showed that that k's setting for peak performance depends on the characteristics of the application domain. CNN's behavior should be highly similar. The latter two questions remain unanswered, although their solutions must also account for domain dependent variables.

The set of instances retained by CNN is dependent on the ordering of the training sequence. Consider the ordering of the training instances shown in Figure 7.1, where the line in the center of the figure delineates positive from negative instances. The instances

are labeled according to their relative distance from the concept boundary. For example, instance number 1 is located furthest form the boundary. If these eight instances comprised the training set and had this ordering, then CNN would save only instances 1 and 2. If they had the reverse ordering, then CNN would save only instances 7 and 8. The two instances saved in both cases are *consistent* with all of the training instances (i.e., the saved instances correctly classify all eight instances using the nearest neighbor prediction function). Also, the number of instances retained by CNN is not always optimal. If the first three instances in the sequence were 7, 2, and 8, then all three would be saved. An algorithm that discarded currently useless instances would help to further reduce CNN's storage requirements. Gates (1973) examined such an algorithm, which he called the *reduced nearest neighbor* (RNN) rule. RNN cycles through the stored instances, testing to see whether their removal introduces classification errors for the original training set. If not, then the instance is removed. This continues until none of the stored instances are removed during a cycle. Gates applied several variants of both algorithms to Sir Ronald A. Fisher's (1936) iris database and found that RNN always reduced CNN's storage requirements and its accuracy on test instances was at least as good as CNN's in all but one of the fourteen experiments. However, Gates reported that RNN usually required more than twice the amount of time to execute. Explorations of the tradeoff between RNN's additional computational costs and its potential increase in predictive accuracies have not been published.

Kurtzberg (1987) also used a variant of the CNN and IB2 algorithms in his application to a handwritten character recognition task. His algorithm is identical to IB2 except that it employs a domain-specific normalization function, it is not defined for symbolic and missing attribute values, and it uses domain-specific parameters in its similarity function. Similarity is defined as the city-block distance (i.e., Minkowskian metric with $r = 1$) between two instances, which were represented by seven numeric-valued attributes that were carefully chosen for the application. These attributes are normalized to allow for the comparison of handwritten symbols that were written at different rates. Attribute_difference yields a Boolean value, which depends on whether the difference of the two numeric values is less than a pre-specified threshold difference. These value thresholds are are attribute-dependent. A second set of thresholds, which denote the limit to which characters could be stretched and shrunk in comparisons, was used to minimize the number of costly instance comparisons.

Kurtzberg trained his algorithm with several different threshold settings and also did a lesion study to determine whether the reduced set of attributes improved the performance of his algorithm. The training set consisted of four sets of 72 target concepts (i.e., handwritten characters). The algorithm's accuracy on a separate test set of 288 characters was 99.0%.[2] Also, it saved only 22.7% of the training instances after saving the first set of 72 characters. The lesion test showed that the reduction to the small set of seven attributes did not alter the program's accuracy or storage requirements, but it greatly decreased its computational costs, measured as the amount of instance-matching required during training. Stricter thresholds

[2]Unfortunately, comparison studies with alternative algorithms were not performed.

Table 7.1: Upper bounds on the error rates for the k-NN and Wilson's edited k-NN algorithms for three settings.

k	k-NN	Edited k-NN
1	$2P_B$	$1.2P_B$
3	$1.3P_B$	$1.15P_B$
5	$1.2P_B$	$1.10P_B$

on attribute-value matches decreased storage requirements and the amount of time required to match instances, but also decreased classification accuracy.

In summary, Kurtzberg showed that the IB2 algorithm can exploit domain-specific knowledge using value-matching thresholds and carefully selected attributes. This additional information greatly increased his algorithm's efficiency. Section 7.2 describes other ways in which IBL algorithms can use domain-specific knowledge to improve their efficiency.

7.1.3 Storing Only Correctly Classified Instances

Only a few descendants of Hart's (1968) condensed nearest neighbor algorithm have been examined in the pattern recognition literature. However, there still has been a burst of activity on edited nearest neighbor algorithms. Wilson (1972) began a second line of interest in these algorithms by detailing proofs of convergence for an algorithm that (1) removes training instances that were *incorrectly* classified by k-NN using the other training instances and (2) uses 1-NN to classify test instances with the remaining training instances The purpose of the editing algorithm is to eliminate instances that are a minority among their k-nearest neighbors so that the 1-NN can obtain classification accuracies closer to the Bayes optimal rate. Wilson showed that the error rate for this editing algorithm, with $k = 3$, is lower than for the 5-NN algorithm. Table 7.1 displays the upper bounds on the error rates for the k-NN and Wilson's edited k-NN algorithms for three of k's settings. As shown in the table, the edited k-NN algorithm's error rate more closely approximates the Bayes optimal rate (P_B) for smaller values of k. However, this does not imply that better results will always be obtained by using Wilson's editing algorithm because his analysis assumes an infinite number of noise-free instances are available for training.

Wilson's work inspired a flurry of mathematical analyses of his edited k-NN algorithm. For example, Wagner (1973) showed how Wilson's lengthy proofs can be reduced to one page in length. Penrod and Wagner (1977) later showed that Wagner's and Wilson's proofs were incorrect; they had mistakenly assumed that the instances saved after the editing process were uniformly distributed. By restricting the instance space's dimensionality to one and by restricting the selected nearest neighbors of an instance x to have higher values along the predictor dimension, Penrod and Wagner were able to show that Wilson's algorithm

Table 7.2: Upper bounds on the error rates for k-NN, Wilson's (1972) estimates for edited k-NN, and Penrod and Wagner's (1977) revised estimates for Wilson's edited k-NN algorithm for three settings.

k	k-NN	Edited k-NN: Wilson	Edited k-NN: Penrod & Wagner
1	$2P_B$	$1.2P_B$	$1.27P_B$
3	$1.3P_B$	$1.15P_B$	$1.20P_B$
5	$1.2P_B$	$1.10P_B$	$1.17P_B$

Table 7.3: Upper bounds on the error rates for the k-NN, Penrod and Wagner's (1977) estimates for Wilson's (1972) edited k-NN algorithms for 1-dimensional spaces, and Koplowitz and Brown's (1981) modified edited k-NN algorithm for any dimensionality (where $k' = k$) for three settings.

k	k-NN	Edited k-NN: 1-dimensional	Edited k-NN: n-dimensional
1	$2P_B$	$1.27P_B$	$1.27P_B$
3	$1.3P_B$	$1.20P_B$	$1.12P_B$
5	$1.2P_B$	$1.17P_B$	$1.06P_B$

converges slower than was thought to the Bayes optimal rate. The updated convergence rates are about 6% slower, as exemplified in Table 7.2. Koplowitz and Brown (1981) extended this work to n dimensional instance spaces, but had to introduce a somewhat contorted modification of Wilson's original algorithm to guarantee Wilson's implicit assumption that the retained instances are uniformly distributed in the instance space. Their algorithm first partitions samples into groups of k. It then checks to see whether at least $k' \leq k$ instances in each group have the same classification. If so, then *all* of the instances in the group are relabeled with the majority classification. Otherwise, the *entire* group of instances is deleted from the training set. This process highlights the locations of disjuncts. Koplowitz and Brown showed that their modified k-NN editing algorithm speeds the convergence to the optimal Bayes rate, as exemplified in Table 7.3. However, Broder, Bruckstein, and Koplowitz (1985) later learned that Wilson's assumption *does* hold when the dimensionality is large, which they proved based on the fact that, for larger dimensional spaces and sufficient number of training instances, the set of k-nearest neighbors for two instances that are among the set of k-nearest neighbors of a third instance are, with high probability, mutually disjoint except for their own inclusions. Thus, Wilson's results still stand for higher dimensional spaces, but the dimension at which the independence assumption begins to hold is still unknown.

In summary, k-NN editing algorithms speed the convergence rate of k-NN algorithms. A plausible question to ask is whether iterative-editing algorithms would further improve convergence. Penrod and Wagner (1977) showed that, for at least in one dimensional instance spaces, the upper bound on the error rate for one and two iterations of Wilson's (1972) editing algorithm is $1.269P_B$ and $1.162P_B$ respectively for $k = 1$, which indicates that iterative editing schemes might be practical. Tomek empirically tested the iterative algorithm and

the *all-k* algorithm in an application with a one-dimensional instance space. The all-k algorithm applies Wilson's algorithm first with $k = 1$ and repeats with incrementally higher values of k until k is equal to the size of the training set. Tomek found that, while the iterative editing algorithm recorded lower misclassification rates, the all-k editing algorithm recorded the lowest error rates among all the algorithms tested. Devijver (1986) describes similar results with an iterative k-NN editing algorithm and also proves that the upper bound on the fraction of instances edited by his algorithm is bounded above by twice the 1-NN algorithm's error rate on the initial training set.

Another way to obtain higher classification accuracies with k-NN editing algorithms is to require that classifications be based on more than a simple majority vote among the k-nearest neighbors. Tomek (1976b) explored the use of a threshold $\frac{k}{2} \leq k' \leq k$ with Wilson's k-NN editing algorithm. Like Koplowitz and Brown's (1981) algorithm, this modification clarifies the boundaries between disjuncts in the instance space. Tomek's threshold proved useful in an application with a simple artificial domain. He also discussed the utility of similarity thresholds, which is of keen interest to IBL researchers in machine learning (e.g., Bradshaw, 1987; Kibler & Aha, 1988). These thresholds are needed to prevent stored instances from classifying highly dissimilar instances. If a novel instance's k-nearest neighbors are sufficiently similar, then it should be stored. Test instances in these situations are said to be rejected (Hellman, 1970). Tomek described four methods for implementing this option and applied one method to a simple artificial domain with impressive results. However, he left unspecified how these thresholds could be learned.

Dasarathy (1980) provided a solution for learning both of these thresholds. His *neighborhood census rule* (NCR) defines the threshold on the lowest number $k' \leq k$ of instances required by a class c among a novel instance's k-nearest neighbors to differ among classes: it is the smallest number of *acceptable* stored instances in c among any stored instance's k-nearest neighbors. A stored instance in class c is *acceptable* when it is within distance D_c of the instance to be classified, where D_c is the maximum distance between any stored instance y and an instance in class c among y's k-nearest neighbors. Dasarathy's learning algorithm first uses Wilson's k-NN editing algorithm, during which time it learns the class-dependent majority vote and similarity thresholds. It then uses these thresholds in its modified k-NN classification algorithm to classify a set of test instances. The set of "rejected" (i.e., unclassified) test instances are given to an unsupervised learning algorithm for clustering and labeling. Afterwards, this process cycles with these newly-labeled test instances as the training instances. The threshold parameter settings are updated during each cycle. Dasarathy noted that this algorithm is useful when not all of the classes in the instance space are known a priori to training. In such cases, his algorithm tends to classify them as having "'unknown" classifications rather than misclassify them as members of one of the known classes. Dasarathy tested his algorithm on Fisher's (1936) iris database. He ensured that instances in the well-separated class (i.e., Iris Setosa) appeared only in the test set. His algorithm reduced the number of instances misclassified by a k-NN algorithm

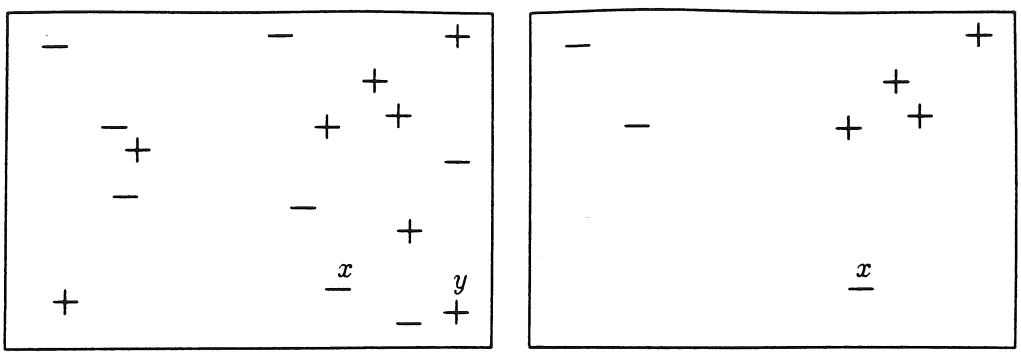


Figure 7.2: An unedited training set (left) and the instances retained after 3-NN editing. Retained instance x would misclassify instance y from the unedited training set.

from 58 to 4. However, this experiment was tailored for Dasarathy's algorithm; the unknown class is linearly separable from the classes represented in the training set. Devijver (1981) pointed out that Dasarathy's algorithm incorrectly assumes that the instances saved by Wilson's k-NN editing algorithm are *consistent* with the instances in the original training set. A subset of a training set is consistent only if its instances correctly classifies all of the instances in the original training set using the 1-NN algorithm. In fact, k-NN editing does *not* guarantee this will occur. For example, if the unedited training set contains the instances in the two-dimensional instance space shown in the left of Figure 7.2, then the retained instances from 3-NN editing are those shown in the right instance space. Negative instance x would misclassify positive instance y from the original training set. Several other of the original training set's instances would also be misclassified by the edited training set. Thus, the 3-NN editing algorithm did not preserve the consistency of the training set. Since Dasarathy's algorithm assumes that k-NN editing preserves a training set's consistency, its applicability to more challenging task domains is still in doubt.

7.1.4 Combined-Editing Algorithms

While Wilson's (1972) editing scheme tends to retain instances that are distant from concept boundaries, Hart's (1968) CNN rule tends to save instances near the boundary between concepts. An algorithm that combines both methods might yield lower storage requirements than would either editing method used alone. Tomek (1976a) examined the behavior of a combined algorithm on an application involving a simple two-dimensional artificial domain. The k-NN algorithm removed most of the boundary instances. Afterwards, the CNN algorithm removed the instances that were on the "interior" of the remaining clusters. The reduction in storage requirements was tremendous (i.e., 53%, 67%, and 11% of the training set was retained by CNN, k-NN editing, and the combined algorithms respectively). Voisin

and Devijver (1987) described two applications of the *Multiedit* algorithm, an iterative variant of Wilson's (1972) k-NN editing algorithm, followed by the CNN rule. Both applications were complicated character recognition tasks involving several type fonts and type qualities. The first application was limited to a single font style, although several different type qualities were represented in the 10,000 instance training set representing 57 categories. The algorithm's error rate on an independent set of 9057 characters was 0.15%, the lowest among the eight algorithms they tested. An optimized perceptron learning algorithm's error rate was twice as high. Voisin and Devijver explained that this occurred because the concepts, which are not unimodal, are also not linearly separable. Furthermore, their algorithm saved only 70 instances (i.e., 0.7% of the training set) to achieve this high performance. In a second experiment, the algorithms were trained on 3780 instances drawn from three type fonts and tested on 8303 separate characters. Their combination algorithm retained only 136 instances (i.e., 3.6% of the training set) and recorded a nearest neighbor misclassification rate of only 0.6%.[3] In summary, combining the two types of editing algorithms resulted in high classification accuracies in a challenging character recognition application.

The instances saved by these combined algorithms are somewhat similar to the ones saved by IB3 and its descendants. As shown in Figure 4.15 on page 87, IB3 tends to discard instances along the boundaries between concepts and retains a set of separating instances closer to the interior of each disjunct. The combined k-NN editing and CNN algorithm does likewise, where the the former removes boundary instances and the latter removes surrounded instances in the interior of disjuncts. However, IB3 differs from this algorithm in that it processes instances incrementally; it does not require a second pass through the set of stored instances.

7.1.5 Summary

In summary, the pattern recognition community has mathematically and empirically analyzed several storage-reduction strategies for instance-based prediction algorithms. Their central concern with editing algorithms is that they approximate the behavior of the Bayes optimal classifier on the original training set. Since instances are assumed to be noise-free, a primary concern is that the edited data set can correctly classify all of the original training instances. Several issues that are of concern in the machine learning community have not been addressed in the literature on edited nearest neighbor algorithms, including the toleration of noisy data, the reduction of dimensionality, PAC-learning analyses, and, in most cases, careful comparison studies with algorithms from other computational frameworks, although Voisin and Devijver's (1987) study is an exception. However, several useful ideas have been proposed and evaluated in this literature that have not yet been implemented in algorithms described in the machine learning literature, including discarding only incorrectly classified instances, combining different editing methods, repeating the editing process, and

[3]Results for the other algorithms were not given for the second experiment.

requiring more than a simple majority among the k-nearest neighbors for correct classifications. Likewise, several ideas developed for supervised learning algorithms have not yet been studied for edited nearest neighbor classifiers, such as incremental learning, the IB3 noise-toleration strategy (Aha & Kibler, 1989), methods for deriving abstractions (e.g., Salzberg, 1990), normalization functions, indexing strategies (e.g., Moore, 1990), and exponentially decreasing similarity functions. Cross-fertilization of these literatures should yield improved IBL algorithms.

Not all of the instance-based algorithms in the pattern recognition literature have been described in this section. Surprisingly, the oldest edited nearest neighbor algorithm closely resembles one of the most prominent IBL algorithms in the machine learning literature and has been applied to similar tasks. For this reason, it is described in the following section.

7.2 Other Instance-Based Learning Algorithms

This section describes a dozen instance-based learning systems that were developed to solve supervised learning tasks. Although most of these systems are well-known in the machine learning literature, some of them were not conceived of as learning algorithms. Nonetheless, all of them can be described by how they instantiate the framework detailed in Chapter 2. The following subsections describe each system's applications, evaluations, and contributions to the IBL framework.

There are a large number of dimensions on which these systems can be compared, contrasted, and organized. Some were grouped according to their principal contributions to the IBL framework. For example, Sections 7.2.1 and 7.2.2 describe systems that use instance-based methods to learn generalizations and process symbolic attribute values respectively. Other systems were grouped to highlight the issues involved with applying IBL algorithms to a specific task. For example, Section 7.2.3 describes IBL systems that solve problems for physical systems. Section 7.2.4 describes two influential systems that use domain specific knowledge to compute similarities. Finally, Section 7.2.5 summarizes the most successfully applied IBL system.

7.2.1 Learning Abstractions of Specific Instances

Instance-Averaging Algorithms

The term *instance-based learning* was first used by Bradshaw (1985) to describe his NEXUS speech architecture's error-recovery process. This system builds a network whose nodes represent primitive acoustic concepts. Raw speech is converted to a usable instance

Table 7.4: Summary of the disjunctive spanning algorithm.

Component	Definition
Pre-processing	Normalization: Dynamic time warping
Performance	Similarity: Euclidean distance Prediction: Nearest neighbor
Learning	Average correctly classified instances Save misclassified instances

representation through a sequence of transformations. First, Raw speech input is transformed by a filter to a digital waveform signal. Fast fourier transforms are then applied to these signals to yield 256-frequency segments from 20 millisecond intervals of speech. These segments are then compressed using 25 filters, whose derivative yields a 25-frequency vector. Finally, a dynamic time warping procedure is used to compare two instances' frequency vectors during similarity computations. NEXUS was applied to the task of learning to recognize spoken letters of the alphabet. The database consisted of two speakers' 30 different pronunciations of the 26 letters. Bradshaw (1987) summarized NEXUS's evaluation in continuous training experiments. After the first 25 blocks of 26 letter pronunciations, NEXUS's average classification accuracy was 93% on the last 5 blocks in single-speaker experiments. In comparison, a non-learning speech recognition system recorded an 80% average classification accuracy for the entire database. NEXUS's accuracy dropped to 84% in a limited multi-speaker experiment.[4]

NEXUS's instance-based learning strategy is summarized in Table 7.4. This algorithm, which Bradshaw (1986) called the *disjunctive spanning* algorithm, is identical to IB2 except that it employs a domain-specific normalization function and averages rather than saves correctly classified instances. This averaging process allows a single instance in the instance space to summarize the locations of any number of similar instances. Therefore, it can be viewed as an abstraction of its nearby neighbors in the training set. Bradshaw's application highlighted the need for domain-specific normalization functions for temporal data. However, the instance-averaging process is of even greater interest because it is a domain-independent algorithm. The experiments in Section 5.3 showed that instance-averaging can achieve better performance than instance-filtering algorithms when the target concepts' disjuncts are unimodal, which may characterize speech data.

Bradshaw (1987) postulated the use of a distance threshold to prevent the disjunctive spanning algorithm from averaging together two highly dissimilar instances. This would help prevent the averaging of two instances in separate disjuncts, which can result in a generalized instance that has an incorrect classification. Furthermore, averaging, which is a form of generalization, can also lead to "instances" that cannot exist due to other application-specific constraints.

[4]The non-learning system's performance was not reported for this second experiment.

Table 7.5: Summary of Sebestyen's (1962) adaptive learning algorithm.

Component	Definition
Pre-processing	Normalization: Dynamic time warping
Performance	Similarity: Euclidean distance Prediction: Nearest neighbor
Learning	Average instances that satisfy the threshold Save all other instances

As explained in Section 7.1.3, Tomek (1976a) and Dasarathy (1980), among others, have examined methods for learning and using distance thresholds in edited nearest neighbor algorithms. However, Sebestyen (1962) is the only pattern recognition researcher that described the use of distance thresholds in an *instance-averaging* IBL algorithm. Sebestyen developed his instance-averaging algorithm after gaining experience with applications of the nearest neighbor function to a speaker recognition task whose training set consisted of two sentences at 20-millisecond intervals spoken by 100 subjects. Speech input was converted to instances that were represented by 4 attributes corresponding to 4 aural frequencies. The nearest neighbor algorithm yields a 91.3% accuracy on a disjoint set of test sentences. However, Sebestyen was concerned with the algorithm's large storage requirements. Therefore, he developed the instance-averaging algorithm summarized in Table 7.5. It is surprisingly similar to Bradshaw's (1987) disjunctive spanning algorithm except for the definition of their prediction functions, which differ in two respects. First, Sebestyen's algorithm saves all instances whose nearest stored instance's distance is above a pre-determined distance threshold. Second, previously stored instances with other classifications are ignored during classification attempts.[5] Sebestyen applied his *adaptive learning* algorithm to a speaker recognition task. It was trained on 200 instances and tested on 200 disjoint instances from each of two speakers. The algorithm recorded a correct speaker classification accuracy of 89.4%. Unfortunately, the value for the distance threshold was neither stated nor varied. Also, Sebestyen did not compare the instance-averaging algorithm and the nearest neighbor algorithm on the same task.

One of the weaknesses of Sebestyen's (1962) algorithm is that the value for the distance threshold was pre-determined rather than learned. Kibler and Aha (1988) examined an extension of Bradshaw's (1987) disjunctive spanning algorithm that learns distance thresholds for each target concept. Distance thresholds were defined as the the shortest distance between a pair of instances in a target concept that appear to lie in different disjuncts. Their algorithm initializes thresholds to unreasonably large values, recomputes their values after each classification attempt, stores all instances until after these thresholds stabilize, and applies the thresholds to the entire training set afterwards. Their algorithm also uses Kohonen's

[5]Sebestyen (1962) did not explain the reasons for this decision. It was probably an oversight since his calculations concerning storage requirements assumed that only one stored instance would lie in an area in the instance space equitable with the size of the pre-determined distance threshold.

Table 7.6: Summary of the NGE learning algorithm.

Component	Definition
Pre-processing	Normalization: linear Instance seeding
Performance	Similarity: Doubly weighted Euclidean distance Prediction: Nearest neighbor or hyper-rectangle
Learning	Save misclassified instances Derive hyper-rectangles for all others Update classification records Update attribute weights

(1988) more conservative instance-averaging method to reduce the number of misclassifications in the partial concept description.[6] The adaptive thresholding algorithm decreased the likelihood of storing misclassified instances and increased classification accuracy in 21 of their 24 experiments. Unfortunately, it also substantially increased storage requirements. In summary, thresholds can be learned and can improve the performance for IBL algorithms that average instances. However, more research is needed to determine whether higher storage requirements can be avoided. One method worth considering is to employ a set of *local* thresholds for each target concept, which should reduce storage requirements by setting low threshold distances near concept boundaries and high threshold distances elsewhere. This would allow for more opportunities to average non-boundary instances.

Learning Hyper-Rectangular Abstractions

Many popular machine learning algorithms for supervised learning tasks partition the instance space into hyper-rectangular subspaces (e.g., Quinlan, 1986a; Clark & Niblett, 1989). Salzberg (1988; 1990) recently developed an IBL algorithm that also has this capability. His algorithm, called NGE (*N*ested *G*eneralized *Ex*emplars), is summarized in Table 7.6. The NGE algorithm uses the standard linear normalization algorithm to pre-process instances and seeds its partial concept description by storing a pre-determined number of instances before training on the remainder of the training set. NGE creates hyper-rectangles whenever it uses a stored instance to correctly classify a training instance, at which time it replaces the stored instance with the smallest hyper-rectangle that includes both instances. If a hyper-rectangle was responsible for the correct classification but did not "contain" the instance,

[6]Initially, Kohonen's (1988) instance-averaging algorithm weights stored instances four times more heavily than the instances they classify. These weights increase slowly each time a stored instance correctly classifies a subsequently presented instance. Thus, while Bradshaw's algorithm initially averages liberally and quickly becomes extremely conservative, Kohonen's algorithm ensures that concept instances are never *quickly* averaged far away from their initial location in instance space and that all correctly classified training instances have a non-trivial impact on the formation of the partial concept description.

then it is minimally extended to include the new instance. Incorrectly classified instances are simply saved.[7] Instances can be nested inside hyper-rectangles with different classifications. Thus, NGE can represent exceptional disjuncts without modifying overly general hyper-rectangular abstractions. NGE maintains a classification record with its stored instances and hyper-rectangles as is done in IB3 and its descendants. However, NGE differs from IB3 in that it continuously weights stored instances rather than require that they pass an acceptance test before they can be used to classify subsequently presented instances. NGE's similarity function is

$$\text{Similarity}(x, y, P) = -\text{Strength}(y)\sqrt{\sum_{i \in P} w_i f(x, y, i)^2} \tag{7.8}$$

where P is the set of predictor attributes, Strength(y) is the percentage of stored instance y's accuracy in previous classification attempts, and w_i is an attribute weight similar to the ones used in IB4. Function $f(x, y, i) = x_i - y_i$ when y is a specific instance. If y is instead a hyper-rectangle, then

$$f(x, y, i) = \begin{cases} x_i - y_i^{\text{upper}} & x_i > y_i^{\text{upper}} \\ y_i^{\text{lower}} - x_i & x_i < y_i^{\text{lower}} \\ 0 & y_i^{\text{lower}} \leq x_i \leq y_i^{\text{upper}} \end{cases} \tag{7.9}$$

where y_i^{upper} is the higher-valued and y_i^{lower} is the lower-valued side of the hyper-rectangle whose plane is parallel to the i^{th} axis. NGE updates attribute weights using the function $w_i = w_i \times \text{Weight\_adjust}(\rho)$, where ρ is a parameter that determines the rate at which attributes are adjusted and Weight_adjust(x) = $1+x$ for correct classifications and Weight_adjust(x) = $1-x$ otherwise. Salzberg notes that this function is imperfect; it does not properly converge on optimal attribute weight settings. IB4's algorithm for learning attribute weights is an improvement on this algorithm in that it tends to avoid this problem, but its behavior has not been formally analyzed.

Salzberg (1988; 1990) applied several variants of his algorithm to four database applications. Although his results were promising, they are difficult to interpret. Some of the results reflected single or few runs of NGE rather than averages over several learning trials. Different variants of the algorithm and different parameter settings were used for each experiment. However, NGE's performance appeared to be similar to or higher than the performances of other supervised learning algorithms on the standard databases included in the experimental evaluation.

NGE introduced several contributions that should prove worthy of future study. The central contribution is a demonstration of how IBL algorithms can learn higher-order "exemplars" efficiently. Salzberg showed how the analyses of hyper-rectangles can lend insights

[7]This is a property of the greedy NGE algorithm (Salzberg, 1988; 1990). However, another variant attempts to shrink hyper-rectangles that incorrectly classify instances. Salzberg (personal communication) has since decided to abandon this strategy.

Table 7.7: Summary of MBRtalk's learning algorithm.

Component	Definition
Pre-processing	Transformation of words to a set of instances
Performance	Similarity: Weighted distribution comparisons Prediction: Weighted k-nearest neighbor
Learning	Save all instances Update counts for weights

on the target concept's structure, which is of central concern for explaining the instance-based algorithm's predictions. The use of hyper-rectangles should also significantly decrease storage requirements for large training sets, especially if the hyper-rectangles can be modified to account for mistakes near concept boundaries. The continuous weighting of stored instances and hyper-rectangles is an interesting and less time consuming alternative to using significance tests. NGE's weight-learning algorithm inspired the development of IB4, which improved on the algorithm and extended it to support concept-dependent attribute weight settings.

Although instance-averaging and instance-generalizing IBL algorithms offer several useful alternatives to storing specific instances, they are currently limited to processing numeric-valued attributes. The next section summarizes IBL algorithms that can process symbolic-valued attributes.

7.2.2 Processing Symbolic-Valued Attributes

MBRtalk

Stanfill and Waltz (1986) introduced a similarity algorithm that can process symbolic-valued attributes and used it in MBRtalk, an instance-based system that was applied to word pronunciation tasks.[8] MBRtalk's description is summarized in Table 7.7. MBRtalk was given a database of words that was also used to test the NETtalk system (Sejnowski & Rosenberg, 1987). A word with l letters is represented using l instances. Instances have nine attributes:[9] the four preceding letters, the focus letter, the four following letters, phoneme,

[8]They referred to MBRtalk as a *memory-based reasoning* system to emphasize the strategic importance of memory as the foundation of intelligent reasoning processes. However, all learning algorithms have some form of memory. The distinguishing property of MBRtalk and other instance-based algorithms is that they retain and use specific instances during supervised learning tasks, placing more emphasis on the dynamic derivation of generalizations rather than on deriving statically applied abstractions. In retrospect, IBL algorithms should instead have been named *extensional* or *lazy* learning algorithms, which highlights their distinguishing behavior.

[9]Stanfill and Waltz have investigated using various numbers of attributes.

Table 7.8: MBRtalk represents the word "file" with four instances.

Letters										Phoneme	Stress
-	-	-	-	f	i	l	e	-		f	+
-	-	-	f	i	l	e	-	-		A	1
-	-	f	i	l	e	-	-	-		l	-
-	f	i	l	e	-	-	-	-		-	-

and stress. The last two attributes are the target attributes. The word representation is exemplified in Table 7.8. Each word is transformed to a set of instances by MBRtalk's pre-processing component. The similarity function is detailed in Section 5.2.1 in Equation 5.14 on Page 131. This algorithm weights attributes according to the degree to which they constrain the potential values of the target attribute. When two instances' values for a predictor attribute a differ, their difference is quantified by comparing the distributions of the target attribute's values for two sets of stored instances – those whose value for a corresponds to a's values in the two instances being compared. The experiments in Section 5.2.1 showed that this similarity function, named the *value-difference* metric, did not improve performance on a set of database applications in comparisons to two other algorithms that were considered and rejected by Stanfill and Waltz. However, this similarity function is more appropriate than the other functions for the word pronunciation task. In particular, it is more appropriate than the overlap metric because predictive relevance varies among an instance's predictor fields. Stanfill and Waltz argued that it is also more appropriate than the weighted-distance metric because different values of a predictor attribute differ in their degree of similarity.[10] MBRtalk's prediction function retrieves the $k = 10$ most similar concept instances, weights them according to the inverse of their distance, and sums the weights for each diagnostic class value. Test instances are classified according to the highest-weighted class. The learning component saves all training instances. Since MBRtalk is implemented on the Connection Machine System,[11] large storage requirements do not significantly decrease the speed of the system. MBRtalk was trained on 4438 words with a total of 32,768 letters and tested on 100 other words with 772 letters. The words were randomly selected from a dictionary. MBRtalk's classification accuracy was 43% for word pronunciations and 86% for phonemes.[12] Comparisons with other speech-pronunciation systems were not reported.

Stanfill (1987) described a test using an extended 11-attribute representation. MBRtalk was trained on 131,072 instances and recorded an 88% phoneme classification accuracy on

[10]Unfortunately, Stanfill and Waltz did not publish performance comparisons with these other two similarity functions. Therefore, this assumption has not yet been validated for the word pronunciation task.

[11]Connection Machine is a registered trademark of the Thinking Machines Corporation.

[12]Stanfill and Waltz (1988) later stated that their accuracy rate for phonemes was 92% in the experiments described in (Stanfill & Waltz, 1986), but this figure is at odds with the results reported in the original ACM article.

a separate set of 1024 instances. He also found that MBRtalk's performance did not tolerate greatly decrease when instances were extended with up to seven irrelevant predictor attributes whose values were randomly selected. This is not too surprising since the value-distance similarity function learns relevance weights for each attribute similar to the weights used in the context model (Medin & Schaffer, 1978). However, Stanfill also reported some impressive results concerning MBRtalk's ability to tolerate predictor attribute noise; its classification accuracy did not appreciably deteriorate until 80% of the predictor attributes' values were noisy, where an N% noise level for an attribute a was defined as replacing a's value with a randomly selected value for a randomly selected N% of the training instances. It is possible that few non-noisy instances in this domain are required to achieve relatively high classification accuracies. Nonetheless, these noise-toleration results are still quite impressive. However, MBRtalk's accuracy decreased linearly with the level of target attribute noise, which is similar to IB1's noise-toleration ability.

PRO

Lehnert (1987) also describes an application of IBL techniques to the problem of word pronunciation. She used an activation-based representation for a network of nodes to represent instances. Using a training set of 750 instances, Lehnert's PRO system correctly determined the pronunciations for 75% of 100 test instances. PRO required fewer instances but larger storage requirements to achieve a classification accuracy similar to that achieved by MBRtalk. PRO was never compared with MBRtalk on the entire training set of 4438 words. Lehnert suggested that doing so would be interesting, but perhaps the time would be better spent after PRO was re-implemented on a high-speed parallel processor, which would allow for more reasonable response times with this large database.

JOHNNY

More recently, Stanfill (1988) used the MBR approach to teach itself how to pronounce written words without supervision in a system named *JOHNNY*. This system was given a set of pronunciation rules relating letters with pronunciations and a phonetic vocabulary, but not a written vocabulary. It then uses these rules to generate plausible pronunciations. JOHNNY then searches the phonetic vocabulary for the most plausible alternative, which is assumed to be correct and is memorized. JOHNNY's word-pronunciation accuracy was 93% on 1024 words from a dictionary. This accuracy increased to 96% on a second pass through these words when the newly memorized instances were allowed to participate in classification attempts. Higher accuracies were recorded in experiments where JOHNNY was told (1) whether its predictions are correct and (2) when it was always given the correct answer after mistakes. Classification accuracies in the latter experiment reached nearly 100%

Table 7.9: Summary of the PEBLS learning algorithm.

Component	Definition
Pre-processing	Set attribute value similarities
Performance	Similarity: Weighted distribution comparisons Prediction: Nearest neighbor
Learning	Save all instances

on the final 100 instances in the training set. This demonstrates the utility of using instance-based algorithms as a learning apprentice, which is a topic discussed further in Section 7.2.4.

PEBLS

Cost and Salzberg (1990) recently used a variant of the value-difference similarity function in their PEBLS instance-based learning algorithm, which is summarized in Table 7.9. PEBLS is an amalgamation of Salzberg's (1990) NGE algorithm and MBRtalk (Stanfill & Waltz, 1986). It was developed specifically to solve supervised learning tasks when the attributes are all symbolic-valued. PEBLS uses the same pre-processing function as MBRtalk for the NETtalk pronunciation task (Sejnowski & Rosenberg, 1987). PEBL's similarity function is

$$\text{Similarity}(x, y, P) = -\text{Strength}(y) \sum_{i \in P} \text{Attribute\_difference}(x_i, y_i, i)^r, \tag{7.10}$$

where

$$\text{Attribute\_difference}(a, b, i) = \sum_{j \in C} \left| \frac{C_{a_j}}{C_a} - \frac{C_{b_j}}{C_b} \right| \tag{7.11}$$

where C is the set of target concepts, C_a is the number of previously stored instances that had value a for attribute i, and C_{a_j} is the subset of C_a whose value for the target concept is j.[13] The value of r was varied depending on the application. Strength weights are defined as the inverse of an instance's accuracy. The first instance's weight is initialized to 1/1. Subsequent instances' weights are initialized to the weight of their nearest neighbor. PEBLS's prediction function, unlike NGE's and MBRtalk's, is simply the nearest neighbor function. Finally, PEBLS saves all training instances in its partial concept descriptions. PEBLS's contribution is a demonstration that variants of the value-difference similarity function can be used in applications other than the NETtalk pronunciation task. Cost and Salzberg reported that PEBLS recorded the highest average classification accuracy for Qian and Sejnowski's (1988) protein folding application and accuracies as high as KBANN's (Towell, Shavlik, & Noordewier, 1990) in their promoter sequence database. In summary, PEBLS performance

[13]PEBLS requires two passes over the training set. It records all the data used in Equation 7.11 during the first pass and uses this information during the second pass.

was as good as or better than other algorithms, including several connectionist networks, for these two applications. Therefore, it appears that the utility of the value-difference similarity function is not limited to applications with the NETtalk database (Sejnowski & Rosenberg, 1987).

7.2.3 Learning to Control Physical Systems

Instance-based algorithms are excellent choices for solving learning tasks in which control knowledge must be learned for dynamic physical systems. These applications require that learning algorithms respond and learn in real time. All IBL algorithms can learn quickly simply by storing newly presented instances. An IBL algorithm can also support quick response times by either (1) restricting the number of instances stored in its partial concept description, (2) using smart indexing techniques which discard old and less accurate instances, or (3) by reducing the number of attributes consulted during classification attempts. This section describes three IBL algorithms that employ these respective methods for decreasing the time required to derive predictions.

Balancing A Pole

CART (Connell & Utgoff, 1987) is an IBL algorithm that employs the first alternative mentioned above: it significantly reduces the number of instances stored. CART was applied to the cart-and-pole problem (Michie & Chambers, 1968; Selfidge, Sutton, & Barto, 1985), where the objective is to balance a vertically-placed pole on a cart traveling along a one-dimensional track of bounded length. Instances are defined in a four-dimensional space whose numeric-valued attribute dimensions are

1. the position of the cart on the track,
2. the velocity of the cart,
3. the angular position of the pole, and
4. the angular velocity of the pole.

The learning algorithm is given a new training instance in each time step (i.e., 0.02 seconds). A learning trial begins with the pole balanced near vertical with the cart centered along the track. The trial ends after the pole fall or after it has been balanced for a satisfactorally long period of time (e.g., Connell and Utgoff chose 5000 time steps). The operation to be controlled is the choice of direction in which to push the cart at each time step. The task involves prediction of numeric rather than symbolic values; each stored instance is labeled with its degree of desirability for balancing the pole. The CART algorithm, which is summarized in Table 7.10, is told that the initial upright position is desirable (i.e., has a

Table 7.10: Summary of CART's instance-based learning algorithm.

Component	Definition
Pre-processing	Normalization: none
Performance	Similarity: Euclidean distance Prediction: Weighted k-nearest neighbor
Learning	Save up to 2 instances per learning trial

target value of 1). If the pole falls during a learning trial, then the instance/state that existed when the pole's angle from vertical exceeded 12 degrees is chosen as a negative instance and is stored with the target value -1. If the pole remained standing for at least 100 time steps, then a heuristic is used to select a relatively good instance that occurred during the learning trial, which is stored with desirability value 1. The prediction function decides which position to push in the current time step by comparing the degree of desirability of the current and previous time steps. This value is calculated using

$$\text{Desirability}(x) = \frac{\sum_i^n Weight(x, y_i) y_t}{\sum_i^n Weight(x, i)}, \tag{7.12}$$

where n is the number of stored instances, y_t is stored instance y's target value (i.e., degree of desirability), and weights are found by computing

$$\text{Weight}(x, i) = \prod_{j \neq i, j=1}^{n} \text{Distance}(x, y_j)^2, \tag{7.13}$$

where the Distance function computes the Euclidean distance between stored instance y_j and the newly presented instance x. The effect of this function is to give more weight to highly similar instances when computing target value predictions. If the computed degree of desirability increased from the previous time step to the current, then the action used for the last time step (i.e., either push left or push right) is repeated for the current time step. Otherwise, the opposite action is chosen. CART's results were excellent; in 14 separate experiments, it always balanced the pole within 18 learning trials. Since at most two instances are stored for each trial, it needed to save at most 34 instances before balancing the pole indefinitely. The fewest number instances stored by CART was 10. In comparison, previous approaches for solving this problem required between 75 and 10,000 learning trials.

CART's main contributions to the IBL framework are a demonstration of how to learn to control a dynamic physical system and, more specifically, the realization that *a strict domain-specific critic can be used to carefully choose and limit storage requirements.* However, this method is insufficient if larger magnitudes of instances are required to achieve good predictive accuracy. CART's strategy is also insufficient for more challenging target functions that exhibit concept drift (Schlimmer & Granger, 1986) since CART assumes that the target function does not change over time.

Table 7.11: Summary of Moore's (1990) learning algorithm.

Component	Definition
Pre-processing	Normalization: linear k-nearest neighbor smoothing
Performance	Similarity: Euclidean distance Prediction: Thresholded nearest neighbor
Learning	Store all instances Discard old and less accurate instances

Learning to Control Motor Behavior

Instance-based learning strategies must employ a smart indexing strategy to support fast response times when training sets are large. They must also accept more instances and continuously remove old and less useful instances when the target concept drifts over time. Moore (1990) combined these strategies in his algorithms for learning robotic control information. Moore's algorithm, which is summarized in Table 7.11, stores instances in a k-d tree (Samet, 1990; Sproull, in press). As mentioned earlier in Section 4.2.1, this strategy significantly decreased the time required to determine an instance's nearest neighbor. Instances in Moore's applications are represented by ⟨state,action,behavior⟩ triples, where the state describes the position and velocity of a robotic arm from the perspective of the robot's retina, the action is vector of joints applied to the state, and the behavior is the observed acceleration of the arm on the retina. Since the applications are subject to noise, Moore also stored an additional "smoothed" behavior value with each state that is calculated using a weighted k-nearest neighbor function on all the instance's neighbors within a pre-specified distance. Storage requirements were minimized by discarding old instances and those whose smoothed behavior value is sufficiently different from the perceived behavior value. Moore's algorithm can be applied to situations in which the state and requested behavior are known while the action is the target attribute. In these cases, the nearest stored neighbor is located using the instance's state and smoothed behavior as the predictor attributes. However, if the nearest neighbor's distance is larger than another pre-specified threshold, then a procedure is used to prevent the choice of poor actions and encourage the robot to experiment with promising alternative actions. Moore (1990) showed that his algorithm can learn to guide a doubly-jointed robotic arm along a sphere, even in the presence of noise and drastic concept shift. The algorithm was also applied to the more challenging task of batting a thrown ball into a bucket, where distances were measured along a single dimension. In this case, the learning task was partitioned into three subtasks: (1) predicting when the ball must be hit, (2) computing an acceleration of the arm to the ball, and (3) positioning the bat appropriately to strike the ball with the correct force to return it into the bucket. Each subtask can be solved using an application of Moore's algorithm. As in the first experiment, information such as the location of the end of the arm (i.e., where

the bat is located), the location of the bucket, the location and velocity of the ball, and the ball's landing location are all given to the learning system. All three subtasks can use the same representation used previously for instances, but the tasks use the information in an associative manner. That is, in some cases the target attribute is the behavior while in others it is the action. Moore's algorithm was able to learn to bat the ball into a statically located bucket after 5 learning trials. It also learned to bat the ball close to the bucket even if it was randomly relocated after a periodic number of learning trails. Finally, it was able to accomplish this same task even when the speed and direction from which the ball was thrown varied with each trial.

Moore's (1990) main contribution was an *impressive demonstration on how an instance-based approach can be used to solve robotic control tasks.* Moore's algorithm exploits the fact that IBL algorithms can determine when sufficiently new situations occur, at which time experimentation is required rather than duplication of the action stored with the nearest but dissimilar stored instance. Another lesson learned is that *forgetting by removing less accurate instances can support good IBL performance in applications involving concept drift.* Similar processes for removing instances and rules are used in several other learning algorithms in both the machine learning and experimental psychology literatures (e.g., Hintzman, 1986; Schlimmer, 1987a; Fisher, 1987; Minton, Carbonell, Etzioni, Knoblock, & Kuokka, 1987; Markovitch & Scott, 1988; Aha & Kibler, 1989). Finally, Moore's second experiment demonstrates how domain-specific knowledge can be used to structure problem solving into a set of IBL applications.

A Cost-Sensitive Learning Algorithm

Moore's robotic applications involve objects that have a single operation; although there is a choice for how to move the arm, there is no choice among different operations such as walking, moving the arm, or grabbing an object. These operations have different computational *costs* involved with determining their values. Under these conditions, a prudent algorithm would employ a cost-effective attribute evaluation method rather than blindly evaluate all available attributes. That is, control is required to choose which attributes to evaluate based on their cost-effectiveness. Tan and Schlimmer (1990) studied this problem and compared the performance of IB2 with a cost-sensitive variant, which they named *CS-IBL.* This algorithm is summarized in Table 7.12. CS-IBL differs from IB2 in that it limits the number of attributes and instances involved in computing the similarities required to make nearest neighbor predictions. CS-IBL repeatedly selects one stored, cost-effective example and evaluates one of its attributes until the nearest neighbor is located. More specifically, CS-IBL selects the stored instance y that maximizes the ratio of expected match success to cost, defined as

$$\frac{\prod_{i \in P'} Pr(x_i = y_i)}{\sum_{i \in P'} C(i) \times \prod_{k=1}^{i-1} (1 - Pr(x_k = y_k))} \tag{7.14}$$

Table 7.12: Summary of the CS-IBL algorithm.

Component	Definition
Pre-processing	Normalization: linear
Performance	Similarity: Restricted Euclidean distance Prediction: Incremental nearest neighbor
Learning	Store only misclassified instances

where P' is the set of predictor attributes evaluated for y but not for the new example x, $Pr(x_i = y_i)$ is the observed frequency with which the two instances have the same value for attribute i (sorted in decreasing order of $Pr(x_i = y_i)/C(i)$ ratios), and $C(i)$ is the cost required to evaluate predictor attribute i. CS-IBL chooses an inexpensive attribute that has a likely value (i.e., one that maximizes $Pr(x_i = y_i)/C(i)$). This is repeated until the upper bound on the distance from the selected stored example is less than the lower bound for one from any other class. If a misclassified instance's evaluated attributes are identical to those for the misclassifying instance, then additional attributes of the new instances are evaluated until it is *sufficiently* distinguishable from the stored instance, where this difference threshold is determined by a pre-specified parameter setting. Tan and Schlimmer applied IB2 and CS-IBL to a domain in which the objective of the Heath Hero 2000 was to determine how to grasp seven classes of cylindrical objects. CS-IBL reduced the overall computation time from over 2100 seconds to about 100 seconds, but it also doubled IB2's error rate. Experiments with irrelevant attributes increased CS-IBL's computation time 10% and its error rate from 11.6 to 15.5 before convergence. IB2's computation time increased by 50% while its error rate was unaffected. Experiments with a simple artificial domain showed that CS-IBL requires less time to converge, generates more errors before convergence, evaluates fewer attributes, and its computational costs remain relatively constant with the introduction of additional irrelevant attributes (although its errors increase). In summary, the cost-sensitive IBL algorithm decreased costs, incurred more classification errors during training, but also increased learning rate.

Tan and Schlimmer's (1990) experiments are the first evaluation of algorithms that attempt to reduce training costs for supervised learning tasks. Their central contribution to the IBL framework concerns the relationship of the similarity and prediction functions; *the IBL framework should be extended to allow the prediction function to control the behavior of the similarity algorithm* (i.e., choose which attributes and instances for which to compute similarity). Figure 7.3 displays the extended performance component of the IBL framework.

All three algorithms in this section are given domain-specific information to improve their efficiency. For example, CART was given a domain-specific critic to determine which instances to save, Moore's algorithm was told how to decompose a difficult volleying task, and CS-IBL was given the relative costs of its available attributes. This domain-specific

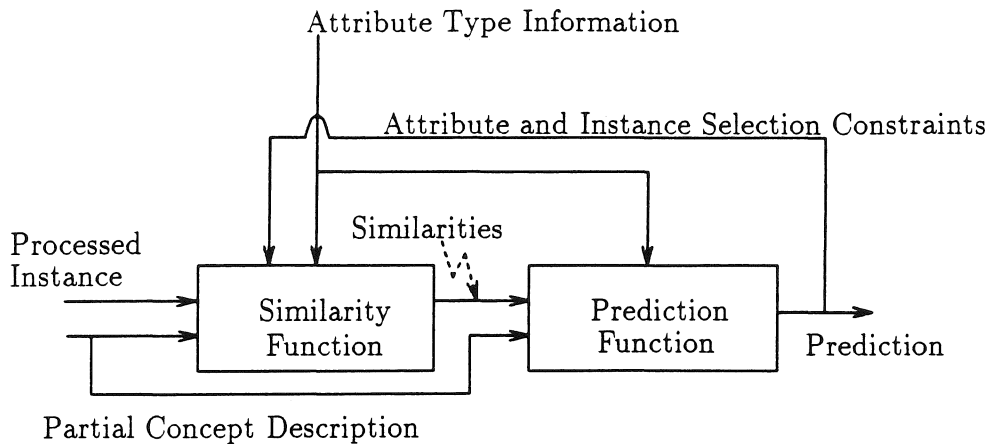


Figure 7.3: Extended performance component for the IBL framework.

Table 7.13: Summary of Optimist's instance-based learning algorithm.

Component	Definition
Pre-processing	Normalization: none
Performance	Similarity: Product of feature similarities derived from a rule-based domain theory Prediction: Weighted k-nearest neighbor
Learning	Stores all instances

information is drawn from an underlying theory that describes the relationships of different objects in the domain. The following subsection summarizes two IBL algorithms that generate explanations of their classifications based on the underlying domain theory.

7.2.4 Using Domain Theories to Explain Classifications

Optimist

Clark (1989) applied an IBL algorithm to the problem of geologic prospect appraisal. The instance-based learning algorithm used in his system, named *Optimist*, is summarized in Table 7.13. Optimist is given interpreted well information (i.e., instances), relevant seismic information (i.e., location of faults), and a domain theory of rules that help to define its similarity function. Optimist's prediction function is a similarity-weighted all-neighbor algorithm. Two instances' similarity is defined as 100 times the product of their attribute

```
Rule for Major Faults::
if fault(F) is between_prospect_and_well
and fault(F) isn't postdepositional
and type of fault(F) is major
then major_fault(f) major fault attribute weight = 0.4
else major_fault(f) major fault attribute weight = 1.0

Distance Rule::
if the distance between a prospect and a well is D kilometers
then Distance(D) modifier = (100-D×5)/100
```

Figure 7.4: Two rules used by Optimist's similarity function.

value similarities, which are determined by a rule-based domain theory. Two of these rules are shown in Figure 7.4. As an example, suppose that a prospecting site lies 5km from a previously explored well and that a major fault lies between them. If the only two attributes defining instances are distance and the presence of intervening major faults, then these instances' similarity is defined as

$$100 \times 0.4 \times (100 - (5 \times 5))/100 = 30 \tag{7.15}$$

Clark calls this an estimate of the *relevance* of the stored instance towards predicting the target values for the prospecting site, which are the site's reservoir thickness and porosity. Clark views the relevance-constructing process as an explanation-generation process. In effect, Optimist's similarity-weighted all-nearest neighbor prediction is based on explanations of the relevance of stored instances for the purpose of predicting the new site's target values. Optimist was delivered to the Enterprise Oil Company in 1989 and has been used regularly on a weekly basis. The eight experts that have used the system remain enthusiastic about its ability to help in the appraise prospective drilling sites.

The main contribution of Optimist is the use of a rule-based theory to dynamically determine instance similarities. Spatial information (i.e., the location of fault lines) is also used to determine these similarities, but it is not specifically stored with any instance. Therefore, *Optimist extends the IBL framework with additional knowledge that is used by the similarity function.* This extension is reflected in Figure 7.5. Optimist's similarity function is similar the one used in the context model (Medin & Schaffer, 1978). Their respective prediction functions also are similar in that they are defined by similarity-weighted k-nearest neighbor functions. However, the context model predicts symbolic target values while Optimist solves numeric prediction tasks. Also, Optimist uses a rule-based theory to set its attribute weights whereas there is no process model for setting the context model's weights. Nonetheless, Optimist's IBL algorithm can be viewed as an instantiated variant of the context model.

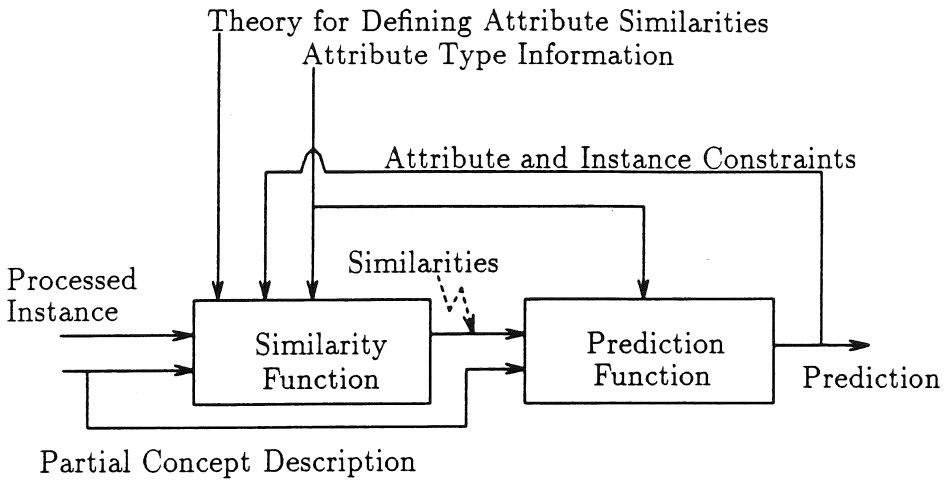


Figure 7.5: The IBL framework's performance component extended to allow domain theories to help derive similarities.

Protos

Optimist uses a rule-based theory to determine attribute similarities. Protos (Bareiss, Porter, & Wier, 1987; Bareiss, 1989a) is another IBL system that uses domain-specific information to determine similarities. Protos is a learning apprentice; it is a knowledge acquisition system that interacts with the user during the normal course of problem solving. Protos is unique among the algorithms described in this section in that it learns a network of features (i.e., symbolic attribute-values), concepts, and exemplars (i.e., possibly generalized instances. Four types of indices are used to relate these objects in the category structure network. First, *remindings* are indices in the network that associate features or exemplars with concepts. Remindings have associated strengths; strong remindings indicate that the feature or exemplar is strongly associated with the corresponding concept. Thus, they can be thought of as attribute weights. Second, *censors* indices indicate that a feature or exemplar is negatively associated with a concept. Absolute censors completely reject potential classifications while weaker censors weaken hypotheses that a new instance will be classified according to the censor's associated concept. Third, *prototypicality* indices link concepts with their stored exemplars. The prototypicality of an instance is defined as a function of its accuracy in previous classification attempts. Finally, *difference* indices are used to distinguish similar instances. These links connect two similar exemplars according to their distinguishing features. The Protos system, which is summarized in Table 7.14, assigns classifications using a two-step process: (1) hypothesis construction and (2) hypothesis confirmation. Step (1) combines remindings and censors associated with the new instance's

Table 7.14: Summary of Protos's instance-based learning algorithm.

Component	Definition
Pre-processing	Normalization: none
Performance	Similarity: attribute-weighted city block derived from explanations of featural equivalence Prediction: nearest neighbor derived from knowledge-based pattern matching
Learning	Merges sufficiently similar instances Stores all others Updates strengths of links in network Adds difference links Updates domain knowledge through user interaction

features to yield a list of possible classifications. The combining process uses nineteen heuristics to explain the relevance of features to category memberships (e.g., two terms which share a common function are likely to be equivalent). Step (2) begins by comparing an instance to the most prototypical instance in the most probable target concept. Their similarity is defined as their number of shared features. However, Protos also constructs an explanation of the shared features equivalence using a process called *knowledge-based pattern matching.* This process locates domain knowledge in the category network that relates the features in the two instances. The domain knowledge consists of a sequence of relations that attempt to equate the two instance's features. Eight types of relations can be encoded in Protos's category structure for this purpose (e.g., generalization relations, part-to-whole relations, causal relations, etc.). Nineteen different heuristics can be used to formulate explanations of the equivalence of features (e.g., the length of a sequence of generalization relationships should not diminish belief in its quality). If unmatched features exist that have high reminding values, then Protos traverses those features' difference links that lead from the stored instance to others that were closely identified with it during past classification attempts. When a sufficiently similar nearest neighbor is located, Protos presents it to the teacher for confirmation. If the teacher rejects the stored exemplar as sufficient for explaining concept membership, then Protos requests the teacher to determine which of its assumptions lead to the incorrect match. Protos then cycles with this new knowledge, repeating the process until the teacher accepts a candidate. At this time, the new instance will either be merged with its nearest neighbor if their similarity is sufficiently high or otherwise stored as a new exemplar in the category network structure. The merging process makes explicit in an exemplar a relationship that was implicit in Protos's domain knowledge, such as replacing features by more general ones. Strength values for remindings, censors, and prototypicality links are updated based on Protos's behavior during the classification attempt. Finally, difference links are inserted between the new exemplar and any that matches that were rejected by the teacher during the classification attempt.

Protos was applied to a clinical audiology database containing 200 training and 26 test instances representing 24 categories. Instances were described by an average of approximately 10 Boolean attributes.[14] Protos retained 120 cases. Its accuracy on test instances was 92.3% (i.e., 24/26) on its first attempt to classify each instance. When directed to attempt a second classification, Protos can correctly classify all 26 test instances, even when no additional knowledge is provided. Two professional clinicians recorded 69% and 85% accuracies on the same test data, while the mean accuracy of 17 graduate students studying clinical audiology was 69%. In summary, Protos's accuracy was better than experts who had spent years studying clinical audiology.

Protos's main contributions to the IBL framework is a *demonstration of how a semantically rich definition for similarity can be used to guide the instance-based learning process.* This is combined with an indexing scheme that greatly reduces the number of instances entering into similarity comparisons. Other novel contributions include the use of censor links (i.e., negative attribute weights) and a detailed demonstration of how users can supply additional domain knowledge to the IBL algorithm.

Several of Protos's innovations are similar to the contributions of other IBL algorithms.

1. Both Protos and CS-IBL (Tan & Schlimmer, 1990) incrementally update the similarity between the new instance and a stored instance through a cyclic process between its prediction and similarity functions.
2. Protos and Moore's (1990) algorithm both use tree-based indexing structures to significantly reduce similarity comparisons.
3. Protos, IB4 (Aha, 1989a), Optimist (Clark, 1989), and NGE (Salzberg, 1990), all use weights to encode attribute relevance information.
4. Like NGE and NEXUS (Bradshaw, 1987), Protos generalizes new instances that are similar to previously stored instances and retains those that are sufficiently different.

However, none of these other algorithms learn anything equivalent to the domain-specific network that Protos uses to relate features, exemplars, and concepts. Of course, other IBL algorithms have unique capabilities that are not supported in Protos. These abilities include:

1. processing numeric-valued attributes (e.g., Kibler, Aha, & Albert, 1989),
2. processing symbolic-valued attributes (e.g., Stanfill & Waltz, 1986),
3. tolerating concept drift (Moore, 1990),
4. tolerating noise and irrelevant attributes (e.g., Aha, 1989a),
5. predicting numeric values (e.g., Clark, 1990), and
6. sensitivity to attribute measuring costs (Tan & Schlimmer, 1990).

[14]This database is characterized by a large percentage of missing attribute values.

Table 7.15: Summary of ALFA's instance-based learning algorithm.

Component	Definition
Pre-processing	Normalization: Standard normal deviation
Performance	Similarity: Attribute-weighted Euclidean distance Prediction: 8-nearest neighbor average – value-difference thresholds decrease search
Learning	Stores all instances
Post-processing	Modifies output using a rule base

Nonetheless, Protos is an excellent example of an exemplar-based learning apprentice. Even though it has only been applied to the domain of clinical audiology, it is highly probable that Protos can be successfully applied to other knowledge-intensive tasks.

7.2.5 Lessons from an Industrial Application

None of the previous eleven IBL algorithms surveyed in this section are being used to support an industrial task on a daily basis, although Optimist receives weekly use. However, ALFA (Jabbour *et al.*, 1987) is used on a daily basis by the Niagra Mohawk Power Company (NIMO) as a load forecasting assistant. NIMO supplies electricity to 3.5 million people in 669 cities, towns, and villages covering approximately 24,000 square miles in New York State. The company employs several experts to predict expected power load requirements for its 109 coal, oil, natural gas, nuclear, and hydro power-generating facilities on an hourly basis. NIMO wastes money when large coal generators and other plants are needlessly activated due to excessively high predictions for power load. Money can also be squandered when excessively low predictions cause the company to buy expensive electricity from other power companies. Thus, millions of dollars can be saved by making accurate power load predictions. ALFA, which was installed in 1987, has helped NIMO to achieve this goal. ALFA is the first automated load forecasting assistant of its kind to be installed in the United States.

Although ALFA is the most successful IBL algorithm to date, measured in its ability to solve a practical problem of interest to a large community, it is a relatively simple algorithm that requires little extension to the IBL framework. ALFA's instance-based algorithm is summarized in Table 7.15. ALFA is given a database of hourly weather data extending over a period of 10 years (i.e., 87,672 instances). Yearly averages are obtained and used to normalize the instances to account for population growth, which increases average power load. ALFA doesn't learn its attribute weights; they were derived from a multivariate linear regression analysis on 15 years of hourly data. Instances are composed of 16 attributes, including hour, day, year, and the target attribute (i.e., numeric power load). The other 12 predictor attributes are three sets of four weather variables obtained from the airports at three cities in New York State. These variables are dry bulb temperature, relative humidity,

wind speed, and opaque cloud cover. ALFA uses the coefficients from the regression analysis in its similarity function – an attribute-weighted Euclidean distance measure. Its predictions are generated using a simple 8-nearest neighbor function. However, as in Protos and CS-IBL, ALFA's prediction function controls the set of stored instances for which similarities will be computed; similarities are computed only for those stored instances whose hour is within one hour and whose day is within one month of the current instance's hour and day. These can be interpreted as value-difference thresholds in the similarity function. For example, any stored instance whose hour value differs by more than 1 with the current instance's hour value are discarded from the potential set of classifying instances. ALFA stores all new instances; it is continually updating its data base.

ALFA unnormalizes the prediction output by its IBL algorithm. However, additional processing is required to account for social factors influencing power load that are not easily derivable by the empirical learning algorithm. These include hourly, daily, and weekly variations due to such factors as working hours and sleeping patterns. Other factors include holidays, special events, major economic events, Christmas shopping and decorative lighting, power plant shutdowns, and plant blackouts. ALFA uses a domain theory of rules to predict absolute offsets based on these factors, which is added to the empirical IBL prediction to derive a final prediction. Thus, ALFA is similar to Protos (Bareiss, 1989a) and Optimist (Clark, 1989) domain-specific knowledge to help make predictions, but it does not integrate this knowledge into its performance component. It instead uses domain knowledge to modify the prediction output by the IBL algorithm. This post-processing step is unique to ALFA.

ALFA's predictions were within 2% average deviation when trained on data collected in during 1970-1983 and tested on the data collected in 1984. This is equivalent to NIMO's experts' accuracy on the same data. ALFA's main benefit is in its efficiency; it requires only twenty seconds to generate daily predictions that require two hours by human experts. Before installing ALFA, NIMO relied on daily rather than hourly predictions from its experts. Since the system inputs weather data collected via satellite to achieve fast response times, NIMO can now update its predictions at any time. For example, NIMO employees use ALFA to generate updated predictions to determine how potential meterological changes might affect power load requirements.

In summary, ALFA (Jabbour *et al.*, 1987) is currently the most successful IBL system. It demonstrated an alternative strategy for amalgamating domain-specific information in which empirical predictions were derived separately from domain-specific influences. This requires an additional post-processing component for the IBL framework, as displayed in Figure 7.6. ALFA also used a value-difference threshold to account for periodic changes in the target function, which limited similarity computations to only the relevant stored instances. Finally, this system is the only one that sets attribute weights based on regression coefficients.

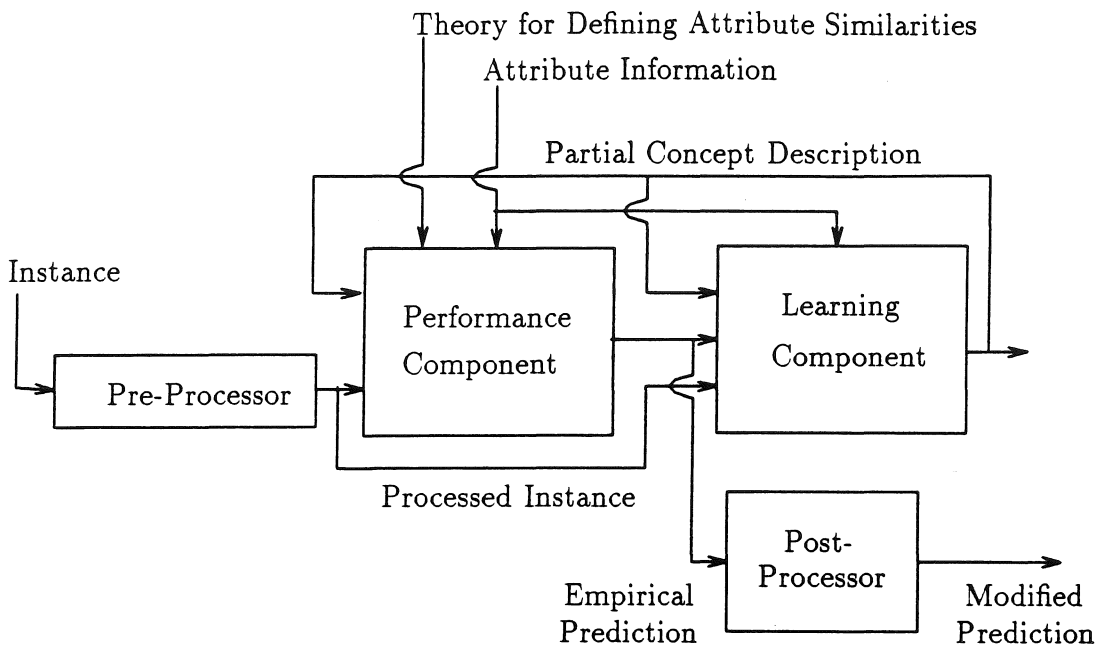


Figure 7.6: Lessons with ALFA suggest that the IBL framework should be extended with a post-processing component.

ALFA's learning component is limited; it doesn't learn new relational information like the category structure used in Protos. It also doesn't learn to update its rule-base or its attribute weights. An interesting question is whether ALFA's general approach, once extended to automate knowledge acquisition, can be used in similar applications.

7.2.6 Summary

Table 7.16 summarizes a major contribution and primary application(s) of the dozen IBL systems surveyed in this section. The contributions refer to novel suggestions concerning the framework described in Chapter 2. The developers of these systems might suggest alternatives and/or additional contributions than the ones shown in this table. However, the purpose of this section is to demonstrate that the IBL framework helps to compare and contrast a large number of inductive learning algorithms that form the basis of a growing paradigm in the machine learning community. All of these systems, with the exception of Sebestyen's (1962) pattern recognition system, were developed within the past six years.

Table 7.16: Summary of the instance-based learning systems.

Name	Major Contribution	Primary Application
NEXUS	Instance-averaging	Speech recognition
(Sebestyen, 1962)	Distance thresholding	Speaker recognition
NGE	Learning hyper-rectangles	Medical diagnosis
MBRtalk	Processing symbolic values	Word pronunciation
PRO	Activation-based network	Word pronunciation
PEBLS	Applications	Molecular biology
CART	Domain-specific storage	Pole balancing
(Moore, 1990)	Indexing/Maintenance method	Robotic motor control
CS-IBL	Cost sensitive prediction	Robotic decision control
Optimist	Domain-specific similarity	Oil prospecting
Protos	Learning apprentice	Clinical audiology
ALFA	Post-processor component	Power load prediction

The algorithms described in this survey, with the exception of Bradshaw's (1987) disjunctive spanning algorithm and Salzberg's (1990) NGE, utilize far more domain-specific knowledge than the IBn algorithms studied in this dissertation. This includes domain specific information concerning how to represent instances, choose which instances to store, determine which instances to discard, assign costs to attributes, and reason about the application domain's theory. The main utility of domain-specific information is to improve the algorithm's learning efficiency. An interesting research problem is to determine how, and under what conditions, can this domain-specific information can be learned without resorting to assistance from experts. The investigation could show many negative results, but the experiments in Chapter 4 showed that domain-independent methods can at least determine domain-specific information such as the predictive effectiveness of stored instances and relative attribute relevance. Other investigations might explore how the benefits of the systems surveyed in this section can be combined without sacrificing efficiency.

Many dimensions for comparing and contrasting the dozen systems have been ignored or not highlighted. In this regard, this survey is incomplete due to its attention to the IBL framework. Also, several other algorithms could have been surveyed. Some were included in Section 7.1's discussion of pattern recognition algorithms. Likewise, some of the algorithms presented here could equally well have been presented in that section. The following section briefly surveys algorithms that, while not properly categorized as instance-based algorithms, depend on the storage of specific instances to achieve their processing goals.

7.3 Using Instances in Other Learning Paradigms

Several algorithms specified by other learning architectures depend on the storage of specific instance information to achieve their goals. Accordingly, this section describes the use of specific instance information in algorithms specified by four learning paradigms that use different representations for concept descriptions: cases, production rules, decision trees, and connectionist networks. The focus of the following discussion is on examples of how algorithms specified by these learning paradigms use specific instance information.

7.3.1 Case-Based Reasoning

Several of the algorithms surveyed in Section 7.2 have been referred to as examples of case-based reasoning (CBR) systems. These include MBRtalk (Stanfill & Waltz, 1986), Protos (Bareiss, 1989a), and PRO (Lehnert, 1987). Case-based reasoning algorithms use previously processed cases (i.e., instances) to solve a new problem case (Rissland, Kolodner, & Waltz, 1989). This definition implies that IBL algorithms are case-based reasoners. However, they address few of the issues of interest to the CBR community. In general, CBR algorithms do the following when processing a new case:

1. recall relevant cases from memory,
2. select the most promising case(s),
3. construct a solution or interpretation of the new case,
4. evaluate the output from (3),
5. apply the retrieved case, and
6. update the memory by storing or integrating the new case with the memory's previously processed cases.

The key issues in case-based reasoning are the development of methods for

1. indexing cases and their solutions properly in memory so that they are retrieved when similar cases are given to the system,
2. selecting relevant stored cases, and
3. adapting solutions stored with previously processed cases so that they can be used with the current case.

While the first list closely resembles the typical processing of instances in the IBL framework, the second list distinguishes their goals. First, indexing and memory organization are of crucial concern to CBR researchers. These issues were completely ignored in this dissertation; instead, my primary interests have been to measure generality, accuracy, learning rate, and

storage requirements in applications to supervised learning tasks. Second, cases in CBR algorithms usually consist of complex structures that are not necessarily representable using a flat set of attribute-value pairs. For example, cases can be plans, problem-solving traces, or even episodes describing a set of interrelated facts. Third, many CBR algorithms operate in problem-solving domains whose problems are not usually viewed as simple prediction tasks. These algorithm store associated solutions and failures with previously processed cases in memory. Retrieved solutions associated with stored CBR cases generally require adaptation so that they can be used to solve the new case. This topic has also been ignored in this dissertation.

I would not argue that CBR algorithms are knowledge-intensive and IBL algorithms are not. Although I focused on investigating knowledge-poor IBL algorithms in this dissertation, the survey in Section 7.2 described several IBL algorithms that use domain-specific knowledge. However, their reasoning mechanisms may differ. For example, IBL algorithms have not yet been designed to reason by analogy, at least from the perspective that analogy requires knowledge-rich graph-matching procedures (Hall, 1989). CBR algorithms can be described as analogical problem solvers.

Nonetheless, there are several reasons to view IBL algorithms as narrowly-focused and memory-intensive CBR algorithms. IBL algorithms address the issues of retrieving relevant cases from memory for solving problems framed as classification tasks. They also address the task of updating memory after classification attempts, at which time instances are stored or generalized and relevant classification information is also updated (e.g., attribute weights, prediction records, etc.). Several of the algorithms surveyed in Section 7.2 further blur the distinction between IBL and CBR. For example, CS-IBL (Tan & Schlimmer, 1990), Protos (Bareiss, 1989a), Moore's (1990) algorithm, NEXUS (Bradshaw, 1985), and PRO (Lehnert, 1987) all address the retrieval problem and attempt to minimize the work involved with classifying a novel case. All but CS-IBL also use intelligent methods for indexing stored cases. Protos additionally addresses the issue of representing failures in the form of difference links. However, none of the surveyed IBL algorithms address the task of adapting retrieved cases because they were designed to solve relatively simple prediction tasks rather than problems that required analogical reasoning.

In summary, although IBL algorithms can be viewed as a proper subset of CBR algorithms, their fundamental focus is on maximizing predictive accuracy while minimizing resource requirements (e.g., time, space) rather than on supporting analogical problem solving in its entirety. The supervised learning task provides IBL with a focus, which allows researchers to more easily evaluate these algorithms empirically. Consider the set of algorithms described in Section 7.2: all of these algorithms have been empirically evaluated in challenging applications. Some IBL algorithms have been mathematically examined to determine their problem solving capabilities (e.g., Kibler, Aha, & Albert, 1989; Aha, Kibler, & Albert, 1990). Others have also been tested for their ability to tolerate domain characteristics that hinder the learning process (e.g., Stanfill, 1987; Aha & Kibler, 1989; Aha, 1989a;

Bareiss, 1989b; Tan & Schlimmer, 1990). Still others have been evaluated for their psychological plausibility (e.g., Aha & McNulty, 1989; Aha & Goldstone, 1990). Most publications on case-based reasoning focus on issues related to the organization and retrieval of cases from memory. Those that do tend to describe single example demonstrations of the methodology proposed, but tend not to describe the rigorous mathematical, empirical, or psychological analyses such as the ones described in Chapters 3, 4, and 6. For example, none of the 69 papers presented at the 1989 Case-Based Reasoning Workshop (Hammond, 1989) described mathematical evaluations while only two (Bareiss, 1989b; Goodman, 1989) described extensive empirical results and only one (Ross, 1989) summarized extensive psychological evaluations of CBR approaches. In fact, *the most thoroughly evaluated CBR systems have all been instance-based learning algorithms that were applied to supervised learning tasks* (Lehnert, 1987; Stanfill & Waltz, 1988; Bareiss, 1989a; 1989b).[15] CBR algorithms are difficult to analyze, which explains why most evaluations of CBR research have been *speculative* rather than *empirical* (Hall & Kibler, 1985). In summary, supervised learning provides a focus that allows IBL algorithms to be empirically evaluated on narrowly constrained problem-solving tasks.

7.3.2 Production Rules

IBL algorithms use similarity functions to yield graded matches between instances. Thus, they can support partial matching. This is a valuable capability for algorithms that learn rules-based concept descriptions (e.g., Michalski *et al.*, 1986; Clark & Niblett, 1989). Without it, rules must match exactly to new instances before they can be applied.

For this reason, Elio & Anderson (1981) extended their ACT rule-based learning system to include partial matching capabilities, which allows ACT to determine the degree of match between its rules and a given instance rather than determine only which ones the instance can instantiate. ACT was proposed as a model of cognitive behavior. Therefore, when Elio and Anderson found that specific instance information was required to simulate behavior displayed by subjects in their experiments, they also extended ACT to save specific instances.

IBL algorithms can be used to reduce training sets so that learning algorithms can derive rules more efficiently. Michalski and Larson (1978) used this approach in their ESEL preprocessor for their AQ11 rule induction algorithm. ESEL's purpose was to reduce training sets to more manageable sizes while still providing AQ11 with sufficient information to yield

[15] One exception to this claim is Fisher's (1987; 1989) evaluations with COBWEB, an incremental algorithm that has primarily been used to solve conceptual clustering problems rather than supervised learning tasks. Fisher argued that COBWEB can be viewed as an efficient implementation of case-based reasoning because it can requires time logarithmic in the number of stored cases to yield classification predictions. COBWEB builds a probabilistic concept hierarchy in the form of a tree whose leaves each contain a set of stored instances. Its hierarchy-updating functions depend on the storage of specific instance information to recalculate conditional probabilities associated with its nodes.

Table 7.17: The ESEL algorithm.

1. Find the distance from each instance to the origin.
2. Locate the instances x and y that are minimally and maximally distant from the origin.
3. Divide the distance d between x and y into r intervals, where r has a pre-specified setting.
4. Partition the training set into r subsets, where the distances to the origin of each instance in the i^{th} subset lie in $[ir, (i+1)r]$.
5. Select the s instances in each subset whose power set product of distances is maximal (i.e., they are maximally distant from each other among the subset).
6. Union these selected instances to form the edited training set.

learn accurate concept descriptions. ESEL selected representative training instances using the algorithm described in Table 7.17. This non-incremental editing algorithm attempts to choose instances that form ever-increasing rings around the origin. Thus, it will work best when the training instances are uniformly distributed around the origin. Later versions of the AQ*nn* series of algorithms (e.g., Michalski *et al.*, 1986) did not employ this pre-processing algorithm, presumably because Michalski and his colleagues learned the same lesson discovered later with algorithms that induce decision trees: post-processing concept descriptions is more robust than pre-processing (Niblett & Bratko, 1986; Mingers, 1989a).

However, Michalski and his colleagues more recently adopted other strategies for their rule-based learning algorithms that are inherent in IBL algorithms. First, Michalski (*et al.*, 1986) introduced a partial matching strategy for AQ15, which helped it to record comparatively good classification accuracies in applications with three medical databases. Michalski (1990) also introduced the concept of lazy generalization into his rule-based learning algorithms. The main thrust of his approach is that concepts can be represented in two parts: a base and a set of inference methods. The base includes assertions describing a rule-based concept description, possibly with additional positive and negative instances. The inference methods are used to dynamically alter the concept according to the context of the current classification task. Thus, Michalski's two-tiered concept description performs limited generalization when training instances are introduced and additional processing during classification attempts. This modification increases the resemblance of the AQ15 system and NGE (Salzberg, 1990). Both systems learn hyper-rectangular exemplars to describe concepts. Both systems also employ partial matching strategies – NGE uses a simple similarity function while AQ15 has a set of matching functions dependent on the relationship between the hyper-rectangle and the instance to be classified. However, Salzberg's algorithm is incremental, learns attribute weight settings, maintains a set of specific instances, and can also process numeric-valued attributes without requiring a *discretization* process that loses specific value information. NGE also recorded higher classification accuracies (i.e., 78% to

68%) for the breast cancer database (Salzberg, 1990; Michalski, 1990), which is the systems' only common application domain.

Part of Michalski's (1990) motivation for retaining a non-incremental rule-based approach for concept induction is due to his belief that IBL systems cannot identify exceptional instances, support context-dependent classifications, record changes to concepts, or store invariant information concerning concepts. Recent work with IBL algorithms have refuted these claims. For example, NGE (Salzberg, 1990) identifies exceptional instances, IB4 (Aha, 1989a) and GCM-ISW (Aha & Goldstone, 1990) support context-dependent classifications, Moore's (1990) algorithm can tolerate concept shift, and several IBL algorithms retain some invariant information in their concept descriptions such as classification records, attribute weights, and higher-order generalizations.

7.3.3 Decision Trees

Algorithms that learn decision trees have a tremendously successful application record (Michie, 1990; Quinlan *et al.*, 1986; Several such algorithms have been described in the machine learning literature, including RuleMaster (Michie *et al.*, 1984), CART (Breiman *et al.*, 1984), ID3 (Quinlan, 1986a) and Assistant-86 (Cestnik *et al.*, 1987) These four algorithms derive decision trees by recursively partitioning the training set based on the selection of a most informative attribute. An attribute test is associated with each interior node n of the tree that partitions the training instances funneled to n. An instance i is classified by following a path through the tree, beginning from the root, whose attribute values match i's until a leaf node l is reached, at which time a classification is made based on a function of the target values corresponding to the training instances that were funneled to l. Recent advances in this area include the development of decision trees algorithms that support the automatic translation to rules (Quinlan, 1987), probabilistic classification (Quinlan, 1990a), efficient multi-attribute splitting techniques (Utgoff, 1989; Utgoff & Brodley, 1990), and constructive induction (Pagallo, 1989; Matheus & Rendell, 1989). Studies of decision tree algorithms have found that they tolerant noise effectively (Quinlan, 1986b), perform better with post-pruning rather than pre-pruning methods (Mingers, 1989a), are relatively insensitive to the splitting algorithm (Mingers, 1989b), and tolerate missing attributes values effectively (Quinlan, 1989).

Schlimmer and Fisher (1986) were the first to publish results on an incremental algorithm for learning decision trees. Their motivation for introducing the incremental ID4 algorithm is the same as Michalski and Larson's (1978) for pre-processing AQ11's training sets: both methods reduce training costs. However, ID4 does not require a separate pre-processing step. Instead, it incrementally updates a decision tree as training instances are processed; it does not need to have at its disposal the complete training set before it builds the tree.

An important characteristic of incremental learning algorithms is that they have low updating costs. However, Schlimmer and Fisher expected that ID4 requires several passes through the data set (i.e., one per level in the tree) to stabilize the tree before an ID3 equivalent tree will be constructed. This prompted the development of ID5R (Utgoff, 1989) which requires only one pass to correctly classify all training instances. ID5R transforms the tree at a node n when an alternative test yields a more informative partition of the training instances funneled to n. The transformation process depends on the storage of specific instances at the leaves of ID5 to support the dynamic re-evaluation of each attribute test's information gain. Van de Velde's (1990) IDL incremental algorithm for constructing minimal-sized decision trees also depends on the storage of specific instance information to achieve its goal.

In summary, algorithms that induce decision trees incrementally use specific instance information to guide the tree-updating transformation process. Instances are used not for the purposes of deriving inductions, but instead for the purpose of improving learning efficiency by reducing updating costs.

7.3.4 Connectionist Networks

Specific instance information also can be used to improve the performance of connectionist algorithms. For example, Volper and Hampson (1987) investigated the use of *specific instance detectors* (SIDs) as additional attributes in multi-layer perceptron networks for learning Boolean functions. They found that adding SIDs can only improve the convergence rate of perceptron training. Furthermore, they found that, if only a limited number of SIDs are available, they are best used to learn the classifications of exceptional instances. Finally, the number of adjustments required for the combined algorithm was found to be empirically lower than when using either SIDs or the base set of features alone for learning several different types of Boolean functions.

Volper and Hampson's work is one of the few on adding links between nodes that are activated for specific instances. At an extreme, their system could delegate one hidden unit per training instance; hidden units would maximize their output for a specific instance. An alternative strategy would (1) link only the input attributes rather than SID detectors to inputs on hidden units and (2) initialize the hidden units to recognize a specific instance. That is, the hidden units' output would be maximized when the input represents their encoded specific instance. Using sigmoidal functions, the hidden units could be designed to activate according to a function of their "distance" from the input instance. If the network has one output node per target class and outputs of each hidden unit H_i activates the class corresponding to its encoded instance and inhibits all other output nodes, then the weights on links between nodes can be set to simulate the nearest neighbor function. This network is summarized in Figure 7.7, which depicts the structure of a network for concept learning

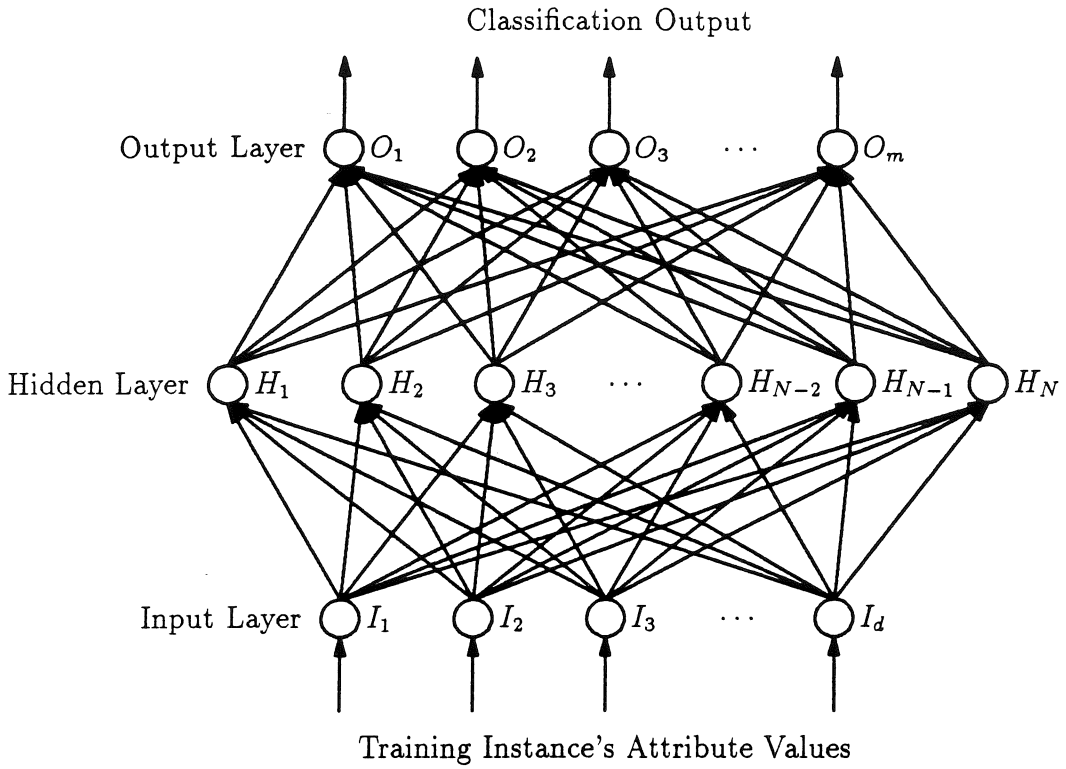


Figure 7.7: Structure for a nearest neighbor network.

when the training set consists of N instances in a d-dimensional instance space for an m-ary classification task. Weights for this multi-layered network can be updated using a feed-forward approach. With appropriate extensions, the k-nearest neighbor algorithm can also be implemented as an artificial neural network (Specht, 1990).

While the nodes of a multilayered perceptron networks learn hyper-planes, the nodes in an instance-based network learn to attend to hyper-polygonal areas in the instance space. Generalizations are determined by interpolating the predictions of hidden-instance nodes in the instance space. If the similarity function decreased exponentially with distance, as it does in the exemplar-based process models described in Chapter 6, then the interpolation is constrained to correctly classify all of the training instances. In effect, this means that the multi-dimensional interpolation mapping from training instances to classifications must *pass through* the training instances. Therefore, k-nearest neighbor networks with an exponential similarity function are a subset of networks whose hidden nodes compute *radial basis functions* (Broomhead & Lowe, 1988). Radial basis function (RBF) networks contain hidden nodes that compute an exponentially decreasing function of the weighted distance of their inputs. Kruschke's (1990) ALCOVE model, which was described in Section 6.3, is a simple example of an RBF network. However, while typical RBF networks encode attentional

strengths in hidden nodes, ALCOVE encodes attentional strengths in the input nodes, which gives it the power of the GCM-MW model (Section 6.4). Attentional weights serve to stretch and shrink the instance space as they do in IB4 (Section 4.4). The activation of ALCOVE's hidden node H_i is computed using the function[16]

$$\text{Activation}(H_i) = e^{-c_i \text{Input\_Distance}(H_i)^2} \tag{7.16}$$

where

$$\text{Input\_Distance}(H_i) = \sqrt{\sum_{j=1}^{d} w_j |H_{i_j} - \text{Activation}(In_j)|^2} \tag{7.17}$$

where c_i is a constant that determines the degree of the exponential gradient, w_j is input node In_j's attribute weight, and H_{i_j} is the attribute value of hidden node H_i for predictor attribute j. The ALCOVE model sets the RBF hidden nodes location to an arbitrary position in the instance space. The Gaussian spheres around an RBF node overlap. Each hidden node responds to a small area in the instance space. Weights from a hidden node to an output/category node O_c indicates the association strength between the nodes region in instance space and category c. The activation at c's output node is simply the sum of the product of these weights with the corresponding hidden node's attentional weight. Weight settings are updated using a gradient descent function.

RBF networks have a guaranteed learning procedure and can explicitly model nonlinear relationships (Broomhead & Lowe, 1988). This contrasts with multi-layered perceptron networks, which learn weights using an unconstrained least squares optimization process, since there is no global convergence theorem for these algorithms. A serious problem with RBF networks is that they tend to overfit the training set. Broomhead and Lowe describe variants of strict RBF networks in which the interpolation constraint is relaxed so that the function need not classify all of the N training instances correctly. One way to relax these constraints is to use fewer than N RBF hidden nodes. Moody and Darken (1988) examined several such RBF networks and used an adaptive k-means clustering algorithm (Kohonen, 1988) to learn locations for a pre-determined number of hidden RBF nodes. While this process learns the centers for the radial basis functions, the network learning process determines their width (i.e., the gradient of their exponential function). Moody and Darken compared their RBF network to the standard back propagation algorithm (Rumelhart *et al.*, 1987) on a challenging time-series prediction task. They found that their algorithm can achieve similar accuracies but requires only 0.01% of the cpu time required by back propagation (BP). They explained that RBF networks learn more quickly because, unlike the hidden nodes in the BP algorithm, their hidden nodes are tuned to only a small portion of the instance space. Also, BP requires three layers of weights while the self-adaptive RBF network requires only two. In summary, BP is slow because it performs a *global* fit to the training data. Self-adaptive RBF networks require only *local* fits. However, these networks

[16]ALCOVE's (Kruschke, 1990) presence parameters for tolerating missing attribute values are ignored here to simplify the discussion.

are limited; they are not appropriate for tasks involving real-time information processing due to their large storage requirements. Nonetheless, significant increases in learning rates result from replacing the sigmoidal functions used in hidden nodes with exponentially decreasing functions.[17] An interesting topic of research on connectionist networks is to investigate whether methods for reducing IBL storage requirements can also be used to reduce the number of hidden RBF nodes required to attain fast learning rates.

One of the drawbacks of the algorithms studied by Kruschke (1990) and Moody and Darken (1988) is that they do not exploit localized attention information as is done in the GCM-ISW (Aha & Goldstone, 1990). However, Volper and Hampson (1990) recently showed that a *focusing* mechanism can support this behavior. A perceptron focuses on a misclassified instance by rotating its weight vector towards the instance by simultaneously decreasing its threshold. Quadratic functions that use this procedure represent localized attention information by stretching and shrinking their localized descriptions. Therefore, their algorithm can simulate the behaviors displayed by GCM-ISW, including asymmetric similarity ratings. However, Hampson and Volper note that an exponential number of hidden nodes may be required to learn arbitrary Boolean functions in a multi-dimensional space, much as storage requirements for IBL algorithms increase exponentially with dimensionality. Again, research on storage reduction in IBL algorithms may be able to contribute useful methods for reducing the number of hidden nodes required in these networks.

Poggio and Girosi (1990) have recently related RBF networks to regularization theory, which assumes that the mapping from inputs to classes is smooth; small changes in the instances cause small changes in the probabilities of classification predictions. Methods in regularization theory select the hyper-surface f that minimizes the function

$$H(f) = \sum_{i=1}^{N}(t_i - f(y_i))^2 + \lambda||Pf||^2, \tag{7.18}$$

where t_i are the hyper-surfaces's target value predictions for the N training instances y_i and λ, the regularization parameter, can be varied to explore trade off between the smoothness of the hyper-surface and its degree of consistency with the training set. Poggio and Girosi prove that multilayer feed-forward RBF networks can be used to find solutions to Equation 7.18. They also describe a self-adaptive method for reducing the number of RBF nodes required that is similar to the algorithms examined by Moody and Darken (1988). These learning algorithms,

[17] In another example, Specht (1990) reported a speed improvement of 200,000 to 1 when switching from BP to an RDF network. The application involved identifying hulls of ships in data donated by the Naval Ocean Systems Center. The RBF networks, run on an IBM/PC at 8 MHz, completed a complete cross-validation study of the 113 instances in the data set in 9 seconds, most of which was spent on screen I/O. Back propagation, run on a DEC VAX 8650 and tuned to use a minimal number of hidden units, required being run over the span of a weekend. BP recorded an accuracy of 82% compared to 85% by the RBF network. Specht guessed that the BP algorithm might attain the RBF network's accuracy if allowed to run for 3 weeks. The RBF network was also run on a PC/AT 386 with a 20 MHz clock. It completed in 0.7 seconds, again with an accuracy of 85%.

which perform task-dependent dimensionality reduction using attentional weights, are similar to instance-based approaches that employ instance-averaging techniques, storage reduction strategies, and learn attribute weights. Poggio and Girosi refer to this general class of RBF network algorithms as *hyper basis functions* and argue that it encompasses several other frameworks for prediction, including k-nearest neighbor algorithms.

As a final note, Wolpert (1990) recently argued that the back propagation algorithm is a poor generalizer. He developed a formal and architecture-independent theory of generalization that relates the underlying semantics of several different learning approaches. He notes that regularization theory comes the closest to matching his generalization theory of surface-fitting, but argues that regularization theory is *too* closely tied to viewing generalization as surface-fitting. That is, regularization theory is a formalism that disregards constraints of what constitutes good generalizations in real-world prediction tasks. According to Wolpert's generalization theory, a generalizer should be invariant under rotations, translations, and scaling of the instance space and also invariant with respect to the ordering of the training instances. Learning algorithms that abide by these constraints include all algorithms that generalize based on interpolations from an instance's k-nearest neighbors. Wolpert showed that, using the overlap similarity function, a similarity-weighted 4-nearest neighbor algorithm records a lower misclassification rate (18%) than BP (22%) on the NETtalk experiment (Sejnowski & Rosenberg, 1987). Furthermore, Wolpert found that an extension of his algorithm that used statically valued attribute weights could reduce this error rate to as low as 7% on the test set. Thus, Wolpert refutes the argument that BP is a good generalizer for the NETtalk database. However, Wolpert does not address whether RBF networks are compatible with his theory of generalization. He states:

> *Nor is there any reason to think that alternative neural net generalization schemes... would perform any better (in comparison...) at these tests.* (page 447)

Quite the contrary; as this survey explains, any of Wolpert's algorithms, including the one he described in his paper, could easily be represented by an artificial neural network that appropriately uses RBF hidden nodes.

In summary, an exciting branch of research on artificial neural networks has recently focused on the use of radial basis functions for hidden nodes in feed-forward networks. The behavior and semantics of these networks is highly similar to IBL algorithms that define similarity to decrease exponentially with distance and use k-nearest neighbor prediction functions. An interesting area of research would be to determine whether variants of storage reduction algorithms used in IBL algorithms might be able to help reduce the number of hidden nodes used in RBF networks.

7.4 Chapter Summary

This chapter reviewed research related to the study of instance-based learning algorithms in the pattern recognition and machine learning literatures. Research on IBL algorithms were predated by the study of edited nearest neighbor algorithms, which began with Sebestyen's (1962) study of an instance-averaging algorithm, which was applied to the task of speaker recognition, and Hart's (1968) condensed nearest neighbor rule, which was applied to the task of typewritten character recognition. Algorithms that retain only misclassified instances as well as algorithms that retain only correctly classified instances have been examined. Research on these algorithms has focused both on empirical studies and on the development of rigorous proofs of increased convergence to Bayes optimal rates. The survey of instance-based learning algorithms in the machine learning literature related a dozen IBL systems by focusing on their contributions to the instance-based learning framework described in Chapter 2. The majority of publications in this literature have been case studies, including applications for power load prediction, oil prospecting appraisal, clinical audiological diagnosis, learning robotic control tasks, word pronunciation, and speech recognition. Most of the systems surveyed fit well in the IBL framework, although some provide suggestions on how it should be extended. Contributions to the framework included cost sensitive similarity and prediction functions, methods for supporting learning apprentice systems, methods for defining similarity for symbolic attribute values, and several methods for learning abstractions. Finally, instances have been used in other learning paradigms to support more efficient learning behavior. For example, there is a growing interest in connectionist networks whose hidden nodes use Gaussian functions rather than linear separators. This class of networks can implement most IBL algorithms.

Chapter Acknowledgements

Dennis Kibler originally inspired me to survey the pattern recognition literature. To my initial surprise, IB2 had essentially been defined and tested 19 years before we first evaluated it in our own empirical tests (Kibler & Aha, 1987; 1990). In retrospect, it would have been surprising had this algorithm *not* been given a thorough treatment elsewhere.

I was fortunate to attend the 1987 MLW workshop, where Ray Bareiss spent many hours demonstrating and explaining the subtle intricacies of Protos and Gary Bradshaw spent time explaining the same for NEXUS. Jeff Schlimmer pointed out that CART (Connell & Utgoff, 1987) is an IBL algorithm. Pat Langley brought MBRtalk to my attention. Jeff and Ming Tan provided details on CS-IBL while Peter Clark did the same for Optimist. Dennis pointed out several subtleties concerning some of these algorithms.

Thanks to Rick Granger for pushing me to explore the literature on artificial neural networks. It turns out that Rick was right: work on connectionist networks *has* contributed to the development of IBL algorithms, including algorithms for processing symbolic values (i.e., MBRtalk). More recently, things have gotten even more interesting; many thanks to Tom Dietterich and Wray Buntine for leading me to the literature on radial basis functions. It appears that this relatively new and exciting research area can tie together many approaches for solving classification tasks. Finally, Steve Hampson provided his usual expert insight on connectionism; I'm fortunate that he is as kind with his time as he is competent in this subject.

Chapter 8
Contributions and Perspective

8.1 Contributions of this Dissertation

The survey in Section 7.2 showed that most investigations on IBL algorithms have focused on their empirical behavior. Several of these were detailed case study demonstrations for supervised learning tasks (Sebestyen, 1962; Stanfill & Waltz, 1986; Jabbour *et al.*, 1987; Kurtzberg, 1987; Bareiss, 1989a; Clark, 1989; Moore, 1990). Some also described comparisons to other algorithms (Bradshaw, 1985; Lehnert, 1987; Connell & Utgoff, 1987; Salzberg, 1990; Cost & Salzberg, 1990). Finally, a few reported more detailed investigations of an algorithm's behavior in one or two application domains (Bareiss, 1989a; Stanfill, 1987; Tan & Schlimmer, 1990).

These empirical evaluations are a good sign for the IBL paradigm; they provide evidence of good performance on a large range of challenging supervised learning tasks. However, these studies can be complimented by the problem-focused empirical evaluations and the alternative types of evaluations that I have discussed in this dissertation, whose primary contributions for clarifying the strengths and weaknesses of the IBL paradigm are as follows:

- *A Framework*: I introduced the first framework for IBL algorithms in Chapter 2 and used it to explain the relationships among and limitations of the four algorithms investigated in Chapters 4 and the algorithms surveyed in Chapter 7. This framework provides a focus for algorithm design; improvements to IBL algorithms can be described by how they instantiate this framework's components, as exemplified by the parametric studies in Chapter 5.
- *Mathematical Analyses*: Chapter 3 describes the first PAC-learning analyses of IBL algorithms. The proofs applied only to IB1, but for any value of k, any dimensionality space, any instance distribution, and for both symbolic and numeric prediction tasks. These analyses helped to determine the generality of IB1 and suggested how its efficiency can be improved (e.g., by saving only near-boundary instances, by reducing dimensionality, etc.).
- *Empirical Analyses*: More elaborate IBL algorithms are not as amenable to mathematical analysis. Thus, the three algorithms introduced in Chapter 4 that *reduce storage*

requirements, *tolerate noisy instances*, and *learn domain-specific similarity functions* were empirically evaluated in rigorous and systematic experiments with artificial domains and several databases. The results showed that these algorithms achieve their goals in both symbolic and numeric prediction tasks. Storage reductions can be obtained by storing only those instances that delineate concept boundaries. Noise can be tolerated by using a significance test to ensure that only predictively accurate instances are used to derive target value predictions. An attribute's predictive relevance can be represented using weights in the algorithm's similarity function that stretch and shrink the interpretation of the instance space independently for each target concept. These strategies all improved the efficiency of the basic IB1 learning algorithm.

- *Parametric Studies*: The first systematic studies of five high-level design choices for IBL algorithms were described in Chapter 5. Different definitions for the pre-processing, similarity, prediction, and learning functions were empirically evaluated in several applications for both symbolic and numeric prediction tasks.
- *Exemplar-Based Process Models*: Chapter 6 introduced four psychologically plausible models of categorization. A survey of the literature indicated that a model's capability to explain psychological phenomena associated with categorization is a function of how well it utilizes attribute correlation information implicit in training instances during prediction tasks. Results from simulations and experiments with the four models added additional evidence to support this claim. This completes the loop relating computer science and cognitive psychology; work on exemplar-based models inspired the development of the IBn algorithms, which in turn inspired details of the GCM-xW process models.
- *Relationship with Other Algorithms*: Chapter 7 detailed the relationship of IBL and edited nearest neighbor algorithms. Relationships with other learning algorithms were also explained, including the close ties with multilayer neural networks that use radial basis functions in their hidden nodes.

In summary, this dissertation described mathematical, empirical, and psychological analyses for instance-based learning algorithms from the perspective of a general paradigm framework.

8.2 The Paradigm in Perspective

Many dissertations in machine learning focus on the presentation and evaluation of a specific system, one which displays sufficiently significant improvements or novel contributions to justify its creation. My dissertation is not of this ilk. Instead, it focuses on a burgeoning learning paradigm, attempting to unite a set of relatively disparate systems sharing a similar thesis but different goals and vocabulary. Therefore, this section focuses on the strengths and weaknesses of the paradigm itself rather than on IB4, which, as its name implies, is but one point in a continuum.

Limitations and benefits always occur with respect to the degree to which some goals are satisfied. The main pursuit of research on IBL algorithms has been to improve learning performance. I will focus on the efficiency performance dimension described in Section 2.4. However, concepts can serve several purposes (Matheus, 1987); other relevant performance dimensions will be considered where appropriate.

8.2.1 Limitations of the Paradigm

Breiman *et al.* (1984) published the most comprehensive critique of instance-based algorithms for learning concept descriptions. The status of these concerns are described in the following list.

1. *IBL algorithms are computationally expensive: they save and compute similarities to all training instances.* Practical IBL algorithms reduce or eliminate this problem by using storage reducing techniques, indexing strategies, parallel processing techniques, and/or smart similarity functions (Stanfill & Waltz, 1986; Bradshaw, 1987; Bareiss, 1989a; Salzberg, 1990; Aha & Kibler, 1989; Moore, 1990; Tan & Schlimmer, 1990). Although Waltz (1990) argues that a powerful parallel processor is required by "memory-based reasoning" algorithms, this is an overly-general statement. In fact, these machines are not required when smart indexing strategies or similarity functions are amenable to the application domain. However, they should always be exploited when available; Stanfill, Waltz, and their colleagues have demonstrated several impressive applications of their algorithms, which fall under the banner of the IBL framework.
2. *IBL algorithms cannot tolerate noise.* Stanfill's (1987) results with the pronunciation task suggest that IBL algorithms can tolerate noise without much extension, but it is unknown whether other algorithms will perform well in this application. Simple IBL algorithms are sensitive to noise, as demonstrated by Breiman *et al.* (1984) and the studies in Section 4.2. However, several strategies (e.g., significance test filtering (Aha & Kibler, 1989), strength weights (Salzberg, 1990), and k-nearest neighbor smoothing (Moore, 1990)) helped IBL algorithms achieve good performance in the presence of noise. These methods can also locate exceptional instances – those that belong to small-sized disjuncts.
3. *IBL algorithms cannot tolerate irrelevant attributes.* Attribute weighting strategies used by IB4 (Aha, 1989a), NGE (Salzberg, 1990), and Protos (Bareiss, 1989a) have supported good performance in the presence of irrelevant attributes.
4. *IBL algorithms are sensitive to the chosen similarity function.* Methods that learn (Stanfill & Waltz, 1986; Aha, 1989a; Salzberg, 1990) or are given (Clark, 1989; Bareiss, 1989a) domain-specific similarity functions have been shown to eliminate this problem. However, these learning methods are limited since they assume that a sufficient function can be found by learning appropriate parameter settings. Also, convergence proofs have not been derived for these algorithms.

5. *IBL algorithms cannot easily tolerate missing attribute values.* The investigation in Section 5.2.1 suggests otherwise.
6. *IBL algorithms cannot easily process symbolic-valued attribute values.* The investigations in Section 5.2.1 and the results obtained by Stanfill and Waltz (1986) and Cost and Salzberg (1990) refute this claim.
7. *IBL algorithms do not convey the structure of the data.* Several IBL algorithms derive abstractions (e.g., Bradshaw, 1985; Bareiss, 1989a; Salzberg, 1990). Protos can build a comprehensive category structure, although most of its information is taught rather than learned. NGE can learn hyper-rectangular abstractions similar to those used in decision trees or rule-based systems.

In summary, all of Breiman *et al.*'s concerns with the IBL approach have been refuted through the construction of algorithms that adopt the benefits of abstraction-oriented systems while still maintaining the basic IBL philosophy of using stored instance information to derive predictions. More elaborate IBL algorithms that learn and maintain abstraction information tend to blur their distinction with abstraction-based learning algorithms (Clark, 1988). Breiman and his colleagues might argue that the resulting algorithms are no longer instance-based. I disagree; these algorithms still derive their predictions based on similarity calculations with a set of exemplars (e.g., possibly generalized instances).

Nonetheless, current-generation IBL algorithms still have several limitations. For example, Branting (1989) argued that the attribute-value representation for instances severely limits the applicability of IBL algorithms; they cannot process higher-order relations or structured attributes (Thompson & Langley, 1989), which are required to represent facts such as legal case histories. IBL algorithms must be given large amounts of domain-specific knowledge to process these types of attributes meaningfully in their similarity functions. This knowledge is usually represented by a graph that details inter-attribute relations. Subsets of these graphs can probably be learned, perhaps by an algorithm that builds concept hierarchies (e.g., Fisher, 1987; Gennari, 1990). Although incremental and non-incremental algorithms exist that can process relational attribute-values (e.g., Iba, Wogulis, & Langley, 1988; Quinlan, 1990b), this remains an unsolved topic for IBL algorithms.

Another problem that plagues IBL algorithms is their restriction to selective induction (Rendell, 1986). Principled procedures for inducing higher-order attributes are necessary if concept descriptions are inherently disjunctive or if the space has high dimensionality, in which case weight-learning strategies such as the one used in IB4 will require more time to learn which attributes are relevant. These situations require representational adjustments so that learning rate improves in a space defined by more relevant predictors (Schlimmer, 1987a; Pagallo, 1989; Matheus & Rendell, 1989). Two particular methods might lead to a solution for IBL algorithms. First, Mehra and Rendell (1989) discuss a method that analyzes inverted instance spaces, where attributes are points in the space and instances are the dimensional axes. The problem is simplified to finding "instances" that lie in different parts of the

space. However, these inverted spaces have high dimensionality, and because the problem of reducing the number of instances in the original space requires a form instance-based learning, the problem is inherently cyclic. Nonetheless, this alternative remains appealing. Another possibility is to maintain logical sufficiency and necessity weights for attributes, which STAGGER (Schlimmer, 1987a) used to constrain the search for predictive higher-order attributes. This method might also assist IBL algorithms.

A third problem with IBL algorithms is that they tend to be needlessly redundant (Clark, 1988); predictions for two similar instances will involve highly redundant similarity computations. This information can be stored to predict values for subsequently presented instances. IBL algorithms implicitly derive predictions using a Voronoi diagram, which partitions the instance space as determined by the similarity function, the prediction function, and the stored instances. Strategies that store and modify these hyper-polygonal partitions and use them in a localized fashion can reduce the number of redundant computations. Since NGE's (Salzberg, 1990) learned hyper-rectangles approximate this information, this problem has been partially solved. IBL algorithms that learn more general hyper-polygonal representations are worthy of future research.

Naturally, the framework presented in this dissertation is also limited; it is too abstract for some algorithms and, as shown in Section 7.2, must be extended to describe some of the more elaborate IBL algorithms. For example, Tan and Schlimmer's (1990) cost sensitive learner, which calculates similarities incrementally, required adding a control loop from the prediction function to the similarity function. Nonetheless, it remains a useful tool since it focuses attention on algorithm design choices and highlights existing problems. However, alternative frameworks were not considered; they may provide better methods for these purposes.

8.2.2 Benefits of the Paradigm

Based on recent advances, the IBL paradigm now appears to be an attractive method for solving supervised learning tasks. Any discussion of their advantages should be prefaced with a note concerning their *relative simplicity*, an opinion that has been echoed in pattern recognition (Gates, 1972), categorization theory (Hintzman, 1984), and machine learning (e.g., Bradshaw, 1987). IBL algorithms represent concepts with a set of possibly generalized instances. Predictions are derived from the target values of similar stored instances. Although more elaborate IBL algorithms now retain abstraction-related information, the basic representation and prediction process remains easy to comprehend.

There are several reasons to retain specific instances. For example, the IBL paradigm offers an alternative to standard expert systems methodology in that rules need not be derived from experts. Instead, *they require only instances*, which Michie (1985) noted as being more easily provided by experts. Of course, all algorithms that learn from examples share this

advantage. However, IBL algorithms differ from most other paradigms in that they *process instances incrementally* and *do not reprocess* previously observed instances. Specific instance storage helps support efficient incremental variants of non-incremental algorithms (Utgoff, 1989; Van de Velde, 1990). It also supports the generation of *precedent explanations* in which similar instances are used to explain predictions. This type of explanation procedure is used by experts in several applications, including telephone traffic switching (Bareiss, personal communication), medical diagnosis (Bareiss, 1989a), power load prediction (Jabbour *et al.*, 1987), oil prospect appraisal (Clark, 1989), and legal reasoning (Rissland & Ashley, 1988; Branting, 1989). Naturally, arguments have also been made for using abstractions to explain predictions (e.g., Michie, 1985). IBL algorithms that learn generalizations (Salzberg, 1990) and abstraction-oriented algorithms that store specific instances (e.g., Utgoff, 1989) have both capabilities, thus blurring the distinction between instance-based and abstraction-oriented learning algorithms.

Storing specific instances reduces information loss, which plagues abstraction-only algorithms. IBL algorithms can solve *ad-hoc queries* (Barsalou, 1983; Kahneman & Miller, 1986), which require predictions for attributes other than the pre-selected target attributes. This is possible because a store of specific instances includes all attribute correlation information in the training set. IBL algorithms can exploit this information to *learn a wide range of probability distributions* for concepts; they are not limited to assuming that instances have a uniform or Gaussian distribution (Fried & Holyoak, 1984). IBL algorithms also can efficiently support *concept drift* (Schlimmer & Granger, 1986; Moore, 1990), where the concept definitions change over time. This is possible because little work is required to update IBL concept descriptions (i.e., simple addition or removal of instances). Finally, specific instance storage allows for the learning of *graded concepts* (e.g., Aha, 1989a) in which the typicality of instances can be exploited to enhance the learning process (Bareiss, 1989a; Clark, 1988; 1989). This capability also allows IBL algorithms to learn overlapping concepts (Aha, 1989b) and recognize relatively novel instances due to their low similarities with stored instances (Dasarathy, 1980; Waltz, 1990). Not surprisingly, this partial matching capability, which is inherent in IBL algorithms, has recently been introduced into abstraction-oriented schemes to improve their learning behavior (e.g., Michalski, 1990; Quinlan, 1990a).

The IBL strategy has certain speed advantages. Computing similarities is an inherently parallel process (Stanfill & Waltz, 1988). Furthermore, smart indexing and control strategies can be used to limit the number of similarity comparisons required to build concept descriptions (Bareiss, 1989a; Tan & Schlimmer, 1990; Moore, 1990). For example, a decision tree algorithm requires on the order of $O(|I|^2 \times |A|^2)$ attribute references to generate an accurate decision tree, where I is the set of training instances and A is the set of attributes used to describe instances (Utgoff, 1989). Updates to variants of k-d trees require an average time of only $O(\log |I|^2)$. Thus, IBL algorithms require only $O(|I| \log |I|^2 \times |A|)$ attribute references to learn concept descriptions (Overmars & Leeuwen, 1982). This can be a significant savings; given 1000 training instances described by 10 predictor attributes, ID3 requires 10^8 attribute references while the IBL algorithm requires less than 2×10^5 references. Both algorithms will

generally classify instances in $O(\log |I|)$ time, but this will not always be true since numeric attributes can be repeatedly used along paths of an ID3 tree and k-d trees are not guaranteed to display $O(\log |I|)$ retrieval times for nearest neighbors (Sproull, in press), although Moore (1990) reported such times for his low-dimensional applications. Also, retrieval times will be higher for values of $k \geq 1$. Nonetheless, properly implemented IBL algorithms have competitive processing times and appropriate heuristics can yield additional improvements (Bareiss, 1989a).

Many abstraction-oriented learning algorithms either discretize or use splitting points for numeric attribute values to increase learning efficiency (Schlimmer, 1987a; Michalski *et al.*, 1986; Quinlan, 1986a). This is useful in decision tree algorithms to counteract its bias for selecting symbolic-valued attributes defined over a large set of values (Cestnik *et al.*, 1987). IBL algorithms process numeric values *without requiring discretization processes* or splitting points. IBL algorithms also have a more *relaxed learning bias* than algorithms that learn hyper-rectangular partitions of the instance space. Symbolic predictions are implicitly obtained from a Voronoi diagram partitioning of the instance space, which allows more general hyper-polygonal representations for concept descriptions as shown in Figures 2.4 and 2.6 in Section 2.3.2. IBL algorithms can achieve faster learning rates when a bias different from hyper-rectangles is required to accurately describe the concept. However, hyper-rectangle partitioners will record faster learning rates in applications that fit their assumptions better than the more general IBL bias.

Finally, IBL algorithms *can support knowledge acquisition.* Although Protos (Bareiss, 1989a) is a prime example of this capability, most instance-based learners abstract some form of empirically-derived knowledge from the training set. For example, IB3 learns which instances should be used to derive predictions while IB4 learns which attributes are relevant to prediction tasks. However, most successful IBL algorithms are given large amounts of domain-specific information. For example, CART (Connell & Utgoff, 1987) is given an efficient instance-selection strategy, Optimist (Clark, 1989) is given domain-specific rules for defining similarity, ALFA (Jabbour *et al.*, 1987) is given attribute difference thresholds and a set of rules for modifying predictions, Moore's (1990) algorithm is given various parameter settings to determine properties such as forgetting rates, and CS-IBL (Tan & Schlimmer, 1990) is told the costs involved with evaluating attributes. My interest has been to determine how much and what types of domain-specific information can be learned rather than taught. These other IBL algorithms demonstrate how to encode domain-specific information and provide a focus for investigating what can be learned efficiently.

8.2.3 The Paradigm *in situ*

IBL is a healthy paradigm for learning. Several researchers have recently explored issues that are of general importance to the machine learning community. These include cost-sensitivity, similarity functions for symbolic attributes, robotic control tasks, domain-specific definitions for similarity, learning apprentices, and industrial applications. IBL algorithms differ from most other learning algorithms in that they have fewer processing demands at presentation time and more processing demands at classification time. However, IBL algorithms that learn large amounts of abstraction-related information tend to blur the distinction with algorithms from other learning paradigms (e.g., consider NGE (Salzberg, 1990) and AQ15 (Michalski *et al.*, 1986)). In fact, the dominant trend in improving the performance of IBL algorithms has been towards increasing their workload when instances are processed and decreasing their workload during classification attempts. The opposite trend reflects much of the recent progress on abstraction-oriented algorithms (Michalski *et al.*, 1986; Volper & Hampson, 1987; Quinlan, 1990a; Michalski, 1990). In most cases, experiments with these hybrid algorithms indicate that they succeed in exploiting benefits of both approaches without greatly reducing learning efficiency. This tends to confirm observations that multiple representations support improved learning algorithms (Utgoff, 1989; Branting, 1989; Matheus, 1987; Michalski, 1990).

Most learning algorithms can be categorized by how they search for accurate concept descriptions. For example, AQ15 (Michalski *et al.*, 1986) uses a specific to general search, TDIDT algorithms (Quinlan, 1986a) conduct a general to specific search strategy, and version space learning algorithms (Mitchell, 1977) use a bi-directional approach. That is, they begin at both ends of a lattice (i.e., the most general and the most specific concepts that can be represented) and move their frontiers towards one another. However, the version spaces method can also be viewed as a general to specific search since it incrementally reduces the number of concept descriptions that perfectly fit the training instances. The search strategy used by IBL algorithms can also be viewed as a general to specific search because storage of additional instances increases the granularity with which the implicit Voronoi diagrams partition the instance space.

8.3 Suggestions for Future Research

The mathematical analyses in Chapter 3 indicate that transformations of the instance space that reduce concept boundary sizes will reduce the number of instances required to learn accurate concept descriptions. Unfortunately, no methods for inducing higher-order attributes yet exist for use in IBL algorithms. Thus, this should be a focus for future research.

Chapter 6 described a psychologically plausible exemplar-based model that exploits localized attribute relevance information. None of the existing IBn algorithms that learn attribute relevance information can support this capability. Extensions of IB4 that learn this information should increase learning rates in applications where attribute relevance varies depending on the location in instance space.

Several IBL approaches have recently been proposed for solving challenging learning tasks. These algorithms have not yet been fully investigated to determine their ability to tolerate additional complexities in the training data. For example, Moore's (1990) algorithm may not produce fast retrieval times in high dimensional instance spaces (Sproull, in press). A strategy for learning relative attribute relevance might be required for these spaces. Therefore, I would like to determine whether IB4 can improve the learning efficiency for robotic controls tasks. Another topic worth investigating is whether higher-order abstractions can be used to focus robotic control tasks on learning general patterns of movement rather than making a series of small adjustments in response to perceived deviations from a goal position. For example, Moore (1990) trained his algorithm to follow a circular path by continually tracking the location of a point that moves along that path. This task would be more easily accomplished if the algorithm could reason that the path is circular. These types of abstractions should improve tolerance for noise. They should also improve the tolerance for higher-order concept shifts, where abstraction-oriented concept description remain useful but require shifting to a different location in the instance space. Other hybrids worth investigating involve determining whether CS-IBL (Tan & Schlimmer, 1990) benefits from strategies that tolerate noise, adapt to concept drift, and learn indexing structures. Several hybrids of IBL algorithms are well worth studying.

IBL algorithms might also be usefully integrated with other types of learning algorithms. For example, they might usefully serve as empirical components in integrated learning systems (e.g., Pazzani, 1989) because they can tolerate noise, tolerate concept shift, and provide typicality ratings. It is possible that problem solving tasks addressed by integrated systems can be decomposed into a set of tasks that can be learned by an IBL algorithm, which is the basic approach that Moore (1990) followed when he trained a robotic arm to bat a thrown ball into a bucket. This might involve the logical *chaining* of IBL predictions; previously generated predictions could be used to help form the input for subsequent prediction tasks. Automated methods for proposing causal relationships between attributes could similarly improve the performance of IBL algorithms, essentially serving as a constructive induction process. Integrated learning systems could also suggest what types of information are worthy of abstracting during prediction attempts rather than derived repeatedly.

Finally, contrary to claims based on empirical evidence with one algorithm from each paradigm (Waltz, 1990; Cost & Salzberg, 1990), many of the algorithms specified by the IBL framework cannot record significantly better predictive accuracies than those representable with multilayer connectionist networks. In particular, those that use radial basis functions for their hidden nodes can simulate simple IBL algorithms such as the IBn algorithms and

MBRtalk (Stanfill & Waltz, 1986). However, these networks have not yet been analyzed empirically, so they may be less efficient than the non-network IBL algorithm implementations. Also, more complex algorithms (e.g., Protos (Bareiss, 1989a)) would be substantially more difficult to implement using a connectionist network. Investigations on the relationships of these two types of algorithms is a useful topic for future research. Of particular interest are investigations for automatically identifying an appropriate number of and locations for hidden RBF nodes (Poggio & Girosi, 1990). Storage reduction strategies used by IBL algorithms might assist in the development of improved RBF network algorithms.

8.4 Closing Remarks

This dissertation investigated whether algorithms that do not maintain abstractions derived from specific instances can support robust learning behavior. I introduced a framework for describing these algorithms and summarized mathematical, empirical, and psychological investigations of algorithms specified by this framework. Although seemingly at a disadvantage, instance-based algorithms are competitive choices for solving supervised learning tasks; several algorithms encompassed by this paradigm performed well in a varied set of challenging research and industrial applications.

Most dissertations on machine learning algorithms focus on one or two types of evaluations. For example, the previous dissertations on instance-based learning algorithms focused on empirical investigations (Bradshaw, 1985; Bareiss, 1989; Salzberg, 1990). However, other types of evaluations are also encouraged in this field, including analyses of mathematical properties and evaluations of cognitive plausibility. I have tried to furnish a first step towards evaluating IBL algorithms along these dimensions under the assumption that others could extend my results in the future. I have also emphasized the empirical study of domain characteristics such as noisy data and irrelevant attributes rather than a more exciting demonstration of achievement in challenging domains such as speech recognition, clinical audiology, and robotic control tasks.

Nonetheless, I hope that it proves useful; I prefer to view my contribution as providing a foundation for understanding the strengths and limitations of the instance-based approach. The framework is simple, pliable, and can relate the algorithms surveyed; it should continue to serve as a means of distinguishing the design choices used in these algorithms. The surveys on edited nearest neighbor algorithms and exemplar-based models should serve as a somewhat biased guide to similar algorithms in pattern recognition and experimental psychology.

Progress on instance-based methodology has attained a level of maturity and accomplishment that places it alongside other valuable paradigms for supervised learning. It is

an exciting and growing topic that is worthy of further study, especially in applications involving robotic decision making, learning apprentices, industrial assistants, and massively parallel computing. Considering its strengths in computational feasibility and psychological plausibility, it should also serve as a means to further progress in cognitive science.

References

Adams, D. (1985). *The more than complete hitchhiker's guide to the galaxy.* London, England: Bonanza Books.

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

Aha, D. W. (1989b). *Incremental learning of independent, overlapping, and graded concepts with an instance-based process framework* (Technical Report 89-10). Irvine, CA: University of California, Department of Information and Computer Science.

Aha, D. W. (1990). *Efficient instance-based algorithms for learning numeric functions.* Unpublished manuscript. University of California, Department of Information and Computer Science, Irvine.

Aha, D. W. (in press). Tolerating noise, irrelevant attributes, and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies.*

Aha, D. W., & Goldstone, R. L. (1990). Learning attribute relevance in context in instance-based learning algorithms. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society* (pp. 141–148). Cambridge, MA: Lawrence Erlbaum.

Aha, D. W., & Kibler, D. (1989). Noise-tolerant instance-based learning algorithms. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 794–799). Detroit, MI: Morgan Kaufmann.

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

Aha, D. W., & McNulty, D. (1989). Learning relative attribute weights for independent, instance-based concept descriptions. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 530–537). Ann Arbor, MI: Lawrence Erlbaum.

Anderson, J. R. (1974). Retrieval of propositional information from long-term memory. *Cognitive Psychology, 6,* 451–474.

Anderson, J. R. (1983). *The architecture of cognition.* Cambridge, MA: Harvard University Press.

Anderson, J. R. (1989). A theory of the origins of human knowledge. *Artificial Intelligence, 40,* 313–352.

Angluin, D., & Laird, P. (1988). Learning from noisy examples. *Machine Learning, 2,* 343–370.

Bareiss, R. (1989a). *Exemplar-based knowledge acquisition.* San Diego, CA: Academic Press.

Bareiss, R. (1989b). The experimental evaluation of a case-based learning apprentice. In *Proceedings of a Case-Based Reasoning Workshop* (pp. 162–167). Pensacola Beach, FL: Morgan Kaufmann.

Bareiss, E. R., Porter, B., & Wier, C. C. (1987). PROTOS: An exemplar-based learning apprentice. In *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 12–23). Irvine, CA: Morgan Kaufmann.

Barsalou, L. W. (1982). Context-independent and context-dependent information in concepts. *Memory & Cognition, 10,* 82–93.

Barsalou, L. W. (1983). Ad hoc categories. *Memory & Cognition, 11,* 211–227.

Barsalou, L. W. (1989). On the indistinguishability of exemplar memory and abstraction in category representation. In T. K. Srull & R. S. Wyer (Eds.), *Advances in social cognition.* Hillsdale, NJ: Lawrence Erlbaum.

Barsalou, L. W., & Medin, D. L. (1986). Concepts: Static definitions or context-dependent representations? *Cahiers de Psychologie Cognitive, 6,* 187–202.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. (1986). Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension. In *Proceedings of the Eighteenth Annual Association for Computing Machinery Symposium on Theory of Computing* (pp. 273–282). Berkeley, CA: Association for Computing Machinery.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. (1989). Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery, 36,* 929–965.

Bower, G. H. (1972). Perceptual groups as coding units in immediate memory. *Psychonomic Science, 27,* 217–219.

Bowyer, A. (1981). Computing Dirichlet tessellations. *Computer Journal, 24,* 162–166.

Bradshaw, G. (1985). *Learning to recognize speech sounds: A theory and model.* Doctoral dissertation, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Bradshaw, G. L. (1986). Learning by disjunctive spanning. In T. M. Mitchell, J. G. Carbonell, & R. S. Michalski (Eds.), *Machine learning: A guide to current research.* Boston, MA: Kluwer Academic Publishers.

Bradshaw, G. (1987). Learning about speech sounds: The NEXUS project. In *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 1–11). Irvine, CA: Morgan Kaufmann.

Bransford, J. D., & Franks, J. J. (1971). The abstraction of linguistic ideas. *Cognitive Psychology, 2*, 331–350.

Branting, L. K. (1989). Integrating generalizations with exemplar-based reasoning. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 139–146). Ann Arbor, MI: Lawrence Erlbaum.

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

Broder, A. Z., Bruckstein, A. M., & Koplowitz, J. (1985). On the performance of edited nearest neighbor rules in high dimensions. In *Institute of Electrical and Electronic Engineers Transactions on Systems, Man and Cybernetics, 15*, 136–139.

Broomhead, D. S., & Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems, 2*, 321–355.

Bruner, J. S., Goodnow, J., & Austin, G. (1956). *A study of thinking.* New York, NY: Wiley.

Buchanan, B., Sutherland, G. L., & Feigenbaum, E. A. (1970). Rediscovering some problems of artificial intelligence in the context of organic chemistry. *Machine Intelligence, 5*, 253–280.

Busemeyer, J. R., Dewey, G. I., & Medin, D. L. (1984). Evaluation of exemplar-based generalization and the abstraction of categorical information. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 10*, 638–648.

California Statistical Software, Incorporated (1984). *Classification and regression trees* [Computer program]. Lafayette, CA: California Statistical Software.

Caplan, L. J., & Barr, R. A. (1988). A comparison of context effects for typicality and category membership ratings. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society* (pp. 538–543). Montreal, Canada: Lawrence Erlbaum.

Carbonell, J. G., Michalski, R. S., & Mitchell, T. M. (1983). An overview of machine learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* San Mateo, CA: Morgan Kaufmann.

Carroll, J. D., & Wish, M. (1974). Models and methods for three-way multidimensional scaling. In D. H. Krantz, R. C. Atkinson, R. D. Luce, & P. Suppes (Eds.), *Contemporary developments in mathematical psychology* (Vol. 2). San Francisco, CA: Freeman.

Cestnik, B., Kononenko, I., & Bratko, I. (1987). ASSISTANT-86: A knowledge-elicitation tool for sophisticated users. In I. Bratko & N. Lavrač (Eds.), *Progress in machine learning.* Bled, Yugoslavia: Sigma Press.

Chase, W. G., & Simon, H. A. (1973). Perceptions in Chess. *Cognitive Psychology, 4,* 55–81.

Clark, P. E. (1988). A comparison of exemplar-based and rule-based concept representations. In *Proceedings of an International Workshop on Machine Learning and Meta Reasoning Logics* (pp. 69–82). Sesimbra, Portugal: Unpublished manuscript.

Clark, P. E. (1989). *Exemplar-based reasoning in geological prospect appraisal* (Technical Report 89-034). Glasgow, Scotland: University of Strathclyde, Turing Institute.

Clark, P. E., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning, 3,* 261–284.

Collins, A. M., & Quillian, M. R. (1972). Experiments on semantic memory and language comprehension. In L. Gregg (Ed.), *Cognition and Learning.* New York, NY: Wiley.

Connell, M. E., & Utgoff, P. E. (1987). Learning to control a dynamic physical system. In *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 456–460). Seattle, WA: Morgan Kaufmann.

Cost, S., & Salzberg, S. (1990). *A weighted nearest neighbor algorithm for learning with symbolic features* (Technical Report JHU-90/11). Baltimore, MD: The Johns Hopkins University, Department of Computer Science.

Cover, T. M. (1968). Estimation by the nearest neighbor rule. *Institute of Electrical and Electronics Engineers Transactions on Information Theory, 14,* 50–55.

Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory, 13,* 21–27.

Dasarathy, B. V. (1980). Nosing around the neighborhood: A new system structure and classification rule for recognition in partially exposed environments. *Pattern Analysis and Machine Intelligence, 2,* 67–71.

Detrano, R., Janosi, A., Steinbrunn, W., Pfisterer, M., Schmid, J., Sandhu, S., Guppy, K., Lee, S., & Froelicher, V. (1989). International application of a new probability algorithm for the diagnosis of coronary artery disease. *American Journal of Cardiology, 64,* 304–310.

Devijver, P. A. (1981). Comments on "Nosing around the neighborhood: A new system structure and classification rule for recognition in partially exposed environments." *Institute of Electrical and Electronic Engineers Transactions on Pattern Analysis and Machine Intelligence, 3,* 696–697.

Devijver, P. A. (1986). On the editing rate of the Multiedit algorithm. *Pattern Recognition Letters, 4*, 9–12.

Dietterich, T. G., & Michalski, R. S. (1983). A comparative review of selected methods for learning from examples. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* San Mateo, CA: Morgan Kaufmann.

Drastal, G., Meunier, R., & Raatz, S. (1989). Error correction in constructive induction. In *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 81–83). Ithaca, NY: Morgan Kaufmann.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis.* New York, NY: Wiley.

Ein-Dor, P., & Feldmesser, J. (1987). Attributes of the performance of central processing units: A relative performance prediction model. *Communications of the Association for Computing Machinery, 30*, 308–317.

Elio, R., & Anderson, J. R. (1981). The effects of category generalizations and instance similarity on schema abstraction. *Journal of Experimental Psychology: Human Learning and Memory, 7*, 397–419.

Feigenbaum, E. A. (1977). The art of artificial intelligence – themes and case studies of knowledge engineering. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (pp. 1014–1029). Cambridge, MA: Morgan Kaufmann.

Fisher, D. H. (1987). *Knowledge acquisition via incremental conceptual clustering.* Doctoral dissertation, Department of Information and Computer Science, University of California, Irvine, CA.

Fisher, D. H. (1988). A computational account of basic level and typicality effects. In *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 233–238). St. Paul, MN: Morgan Kaufmann.

Fisher, D. H. (1989). Noise-tolerant concept clustering. In *Proceedings of the Eleventh International Conference on Artificial Intelligence* (pp. 825–830). Detroit, MI: Morgan Kaufmann.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annual Eugenics, 7*, 179–188.

Fix, E., & Hodges, J. L., Jr. (1951). *Discriminatory analysis, nonparametric discrimination, consistency properties* (Technical Report 4). Randolph Field, TX: United States Air Force, School of Aviation Medicine.

Fix, E., & Hodges, J. L., Jr. (1952). *Discriminatory analysis: Small sample performance* (Technical Report 11). Randolph Field, TX: United States Air Force, School of Aviation Medicine.

Franks, J. J., & Bransford, J. D. (1971). Abstraction of visual patterns. *Journal of Experimental Psychology, 90*, 65–74.

Fried, L. S., & Holyoak, K. J. (1984). Induction of category distributions: A framework for classification learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 10*, 234–257.

Fukunaga, K. (1972). *Introduction to statistical pattern recognition.* New York, NY: Academic Press.

Gams, M., & Lavrač, N. (1987). Review of five empirical learning systems within a proposed schemata. In I. Bratko & N. Lavrač (Eds.), *Progress in machine learning.* Bled, Yugoslavia: Sigma Press.

Gates, G. W. (1972). The reduced nearest neighbor rule. *Institute of Electrical and Electronics Engineers Transactions on Information Theory, 18*, 431–433.

Gennari, J. H. (1990). *An experimental study of concept formation* (Technical Report 90-26). Irvine, CA: University of California, Department of Information and Computer Science.

Gluck, M. A., & Bower, G. H. (1988). From conditioning to category learning: An adaptive network model. *Journal of Experimental Psychology: General, 117*, 227–247.

Gluck, M. A., Bower, G. H., & Hee, M. R. (1989). A configural-cue network model of animal and human associative learning. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 323-332). Ann Arbor, MI: Lawrence Erlbaum.

Goldstone, R. L., Medin, D. L., & Gentner, D. (in press). Attributes, relations, and the non-independence of features in similarity judgments. *Cognitive Psychology.*

Goodman, M. (1989). CBR in battle planning. In *Proceedings of a Case-Based Reasoning Workshop* (pp. 264–269). Pensacola Beach, FL: Morgan Kaufmann.

Hall, R. P. (1985). Differing methodological perspectives in artificial intelligence research. *AI Magazine, 6*, 166–178.

Hammond, K. J. (Ed.). (1989). *Proceedings of a case-based reasoning workshop.* Pensacola Beach, FL: Morgan Kaufmann.

Hart, P. E. (1968). The condensed nearest neighbor rule. *Institute of Electrical and Electronics Engineers Transactions on Information Theory, 14*, 515–516.

Hart, P. E., Nilsson, N., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics*, *4*, 100-107.

Haussler, D. (1986). *Quantifying the learning bias in concept learning* (Technical Report UCSC-CRL-86-25). Santa Cruz, CA: University of California, Computer Research Laboratory.

Haussler, D. (1987). Learning conjunctive concepts in structural domains. In *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 466–470). Seattle, WA: Morgan Kaufmann.

Hayes-Roth, B., & Hayes-Roth, F. (1977). Concept learning and the recognition and classification of exemplars. *Journal of Verbal Learning and Verbal Behavior*, *16*, 321–338.

Hellman, M. E. (1970). The nearest neighbor classification rule with a reject option. *Institute of Electrical and Electronic Engineers Transactions on Systems, Science, and Cybernetics*, *6*, 179–185.

Hintzman, D. L. (1984). MINERVA II: A simulation model of human memory. *Behavior Research Methods, Instruments, & Computers*, *16*, 96–101.

Hintzman, D. L. (1986). "Schema abstraction" in a multiple-trace memory model. *Psychological Review*, *93*, 411–428.

Hintzman, D. L. (1988). Judgments of frequency and recognition memory in a multiple-trace memory model. *Psychological Review*, *95*, 528–551.

Hintzman, D. L., & Ludlam, G. (1984). Differential forgetting of prototypes and old instances: Simulation by an exemplar-based classification model. *Memory & Cognition*, *8*, 378–382.

Hirsh, H. (1990). Incremental version-space merging. In *Proceedings of the Seventh International Conference on Machine Learning* (pp. 330–338). Austin, TX: Morgan Kaufmann.

Holland, J. H. (1986). Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* San Mateo, CA: Morgan Kaufmann.

Homa, D., Sterling, S., & Trepel, L. (1981). Limitations of exemplar-based generalization and the abstraction of categorical information. *Journal of Experimental Psychology: Human Learning and Memory*, *7*, 418–439.

Iba, W., Wogulis, J., & Langley, P. (1988). Trading off simplicity and coverage in incremental concept learning. In *Proceedings of the Fifth International Conference on Machine Learning* (pp. 73–79). Ann Arbor, MI: Morgan Kaufmann.

Jabbour, K., Riveros, J. F. V., Landsbergen, D., & Meyer W. (1987). ALFA: Automated load forecasting assistant. In *Proceedings of the 1987 IEEE Power Engineering Society Summer Meeting.* San Francisco, CA.

Kahneman, D., & Miller, D. T. (1986). Norm theory: Comparing reality to its alternatives. *Psychological Review, 93*, 136–153.

Kearns, M., Li, M., Pitt, L., & Valiant, L. G. (1987a). Recent results on Boolean concept learning. In *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 337–352). Irvine, CA: Morgan Kaufmann.

Kearns, M., Li, M., Pitt, L., & Valiant, L. G. (1987b). On the learnability of Boolean formulae. In *Proceedings of the Nineteenth Annual Symposium on the Theory of Computer Science* (pp. 285-295).

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

Kibler, D., & Aha, D. W. (1988). Comparing instance-averaging with instance-filtering learning algorithms. In *Proceedings of the Third European Working Session on Learning* (pp. 63–80). Glasgow, Scotland: Pitman.

Kibler, D., & Aha, D. W. (1989). Comparing instance-saving with instance-averaging learning algorithms. In D. P. Benjamin (Ed.) *Change of representation and inductive bias.* Boston, MA: Kluwer Academic Publishers.

Kibler, D., & Aha, D. W. (1990). Learning representative exemplars of concepts: An initial case study. In J. W. Shavlik & T. G. Dietterich (Eds.), *Readings in machine learning.* San Mateo, CA: Morgan Kaufmann.

Kibler, D., Aha, D. W., & Albert, M. (1989). Instance-based prediction of real-valued attributes. *Computational Intelligence, 5*, 51–57.

Kibler, D., & Langley, P. (1988). Machine learning as an experimental science. In *Proceedings of the Third European Working Session on Learning* (pp. 81–92). Glasgow, Scotland: Pitman.

Koh, K., & Meyer, D. E. (1989). Induction of continuous stimulus-response relations. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 233-240). Ann Arbor, MI: Lawrence Erlbaum.

Kohonen, T. (1986). *Learning vector quantization for pattern recognition* (Technical Report TKK-F-A601). Espoo, Finland: Helsinki University of Technology, Department of Technical Physics.

Kohonen, T. (1988). An introduction to neural computing. *Neural Networks*, *1*, 3–16.

Koplowitz, J., & Brown, T. A. (1981). On the relation of performance to editing in nearest neighbor rules. *Pattern Recognition*, *13*, 251–255.

Kruschke, J. K. (1990). *ALCOVE: A connectionist model of category learning* (Technical Report 19). Bloomington, IN: Indiana University, Department of Psychology.

Kurtzberg, J. M. (1987). Feature analysis for symbol recognition by elastic matching. *International Business Machines Journal of Research and Development*, *31*, 91–95.

Laird, J., Rosenbloom, P., & Newell, A. (1986). *Universal subgoaling and chunking: The automatic generation and learning of goal hierarchies.* Boston, MA: Kluwer Academic Publishers.

Langley, P. (1987). Research papers in machine learning. *Machine Learning*, *2*, 195–198.

Langley, P., Thompson, K., Iba, W., Gennari, J. H., & Allen, J. A. (1989). *An integrated cognitive architecture for autonomous agents* (Technical Report 89-28). Irvine, CA: University of California, Department of Information and Computer Science.

Lehnert, W. G. (1987). Case-based problem solving with a large knowledge base of learned cases. In *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 301-306), Seattle, WA: Morgan Kaufmann.

Li, M., & Vitanyi, P. M. B. (1989). *A theory of learning simple concepts under simple distributions and average case complexity for universal distribution (preliminary version)* (Technical Report CT-89-07). Amsterdam, Holland: University of Amsterdam, Centrum voor Wiskunde en Informatica.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, *2*, 285–318.

Logan, G. D. (1989). Toward an instance theory of automatization. *Psychological Review*, *95*, 492–527.

Markovitch, S., & Scott, P. D. (1988). The role of forgetting in learning. In *Proceedings of the Fifth International Conference on Machine Learning* (pp. 459–465). Ann Arbor, MI: Morgan Kaufmann.

Markovitch, S., & Scott, P. D. (1989). Information filters and their implementation in the SYLLOG system. In *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 404–407). Ithaca, NY: Morgan Kaufmann.

Matheus, C. J. (1987). *Conceptual purpose: Implications for representation and learning in machines and humans* (Technical Report 87-1370). Urbana, IL: University of Illinois, Department of Computer Science.

Matheus, C. J., & Rendell, L. A. (1989). Constructive induction on decision trees. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 645–650). Detroit, MI: Morgan Kaufmann.

McCarthy, J. (1985). Programs with common sense. In R. J. Brachman & H. J. Levesque (Eds.), *Readings in knowledge representation.* San Mateo, CA: Morgan Kaufmann.

McClelland, J. L. (1981). Retrieving general and specific information. In *Proceedings of the Third Annual Conference of the Cognitive Science Society* (pp. 170–172). Berkeley, CA: Unpublished manuscript.

McDermott, J. (1982). R1: A rule-based configurer of computer systems. *Artificial Intelligence, 19*, 39–88.

Medin, D. L., Altom, M. W., Edelson, S. M., & Freko, D. (1982). Correlated symptoms and simulated medical classification. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 8*, 37–50.

Medin, D. L., Altom, M. W., & Murphy, T. D. (1984). Given versus induced category representations: Use of prototype and exemplar information in classification. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 10*, 333–352.

Medin, D. L., Dewey, G. I., & Murphy, T. D. (1983). Relationships between item and category learning: Evidence that abstraction is not automatic. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 9*, 607–625.

Medin, D. L., & Edelson, S. M. (1988). Problem structure and the use of base-rate information from experience. *Journal of Experimental Psychology: General, 117*, 68–85.

Medin, D. L., & Schaffer, M. M. (1978). Context theory of classification learning. *Psychological Review, 85*, 207–238.

Medin, D. L., & Schwanenflugel, P. J. (1981). Linear separability in classification learning. *Journal of Experimental Psychology: Human Learning and Memory, 7*, 355–368.

Medin, D. L., & Shoben, E. J. (1988). Context and structure in conceptual combination. *Cognitive Psychology, 20*, 158–190.

Mehra, P., Rendell, L., & Wah, B. W. (1989). Principled constructive induction. In *Proceedings of the Eleventh International Conference on Artificial Intelligence* (pp. 651–656). Detroit, MI: Morgan Kaufmann.

Mervis, C. B., & Rosch, E. (1981). Categorization of natural objects. *Annual Review of Psychology, 32*, 89–115.

Meyer, D. E., & Schvaneveldt, R. W. (1971). Facilitation in recognizing pairs of words: Evidence of a dependence between retrieval operations. *Journal of Experimental Psychology, 90*, 227–234.

Michalski, R. S. (1983). A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* San Mateo, CA: Morgan Kaufmann.

Michalski, R. S. (1990). Learning flexible concepts: Fundamental ideas and a method based on a two-tiered representation. In Y. Kodratoff & R. Michalski (Eds.), *Machine learning: An artificial intelligence approach* (Vol. III). San Mateo, CA: Morgan Kaufmann.

Michalski, R. S., & Larson, J. B. (1978). *Selection of most representative training examples and incremental generation of VL_1 hypotheses: The underlying methodology and the description of programs ESEL and AQ11* (Technical Report 867). Urbana, IL: University of Illinois, Department of Computer Science.

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

Michie, D. (1985). *The superarticulacy phenomenon in the context of software manufacture* (Technical Report TIRM-85-13). Glasgow, Scotland: University of Strathclyde, Turing Institute.

Michie, D. (1990). New applications and problems for machine learning. Invited talk, *Seventh International Conference on Machine Learning*, Austin, TX.

Michie, D., & Chambers, R. (1968). Boxes: An experiment in adaptive control. In E. Dale & D. Michie (Eds.), *Machine intelligence 2.* Edinburgh, Scotland: Oliver and Boyd.

Michie, D., Muggleton, S., Riese, C., & Zubrick, S. (1984). Rulemaster: A second-generation knowledge-engineering facility. In *1984 Conference on Artificial Intelligence and Applications.*

Miller, G. A. (1956). The magic number seven plus or minus two: Some limits on our capacity for processing information. *Psychological Review, 63*, 81–97.

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

Mingers, J. (1989b). An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, *3*, 319–342.

Minsky, M. (1983). Why people think computers can't. *Technology Review*, *November/December*, 66–81.

Minton, S., Carbonell, J. G., Etzioni, O., Knoblock, C. A., & Kuokka, D. R. (1987). Acquiring effective search control rules: Explanation-based learning in the PRODIGY system. In *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 122–133). Irvine, CA: Morgan Kaufmann.

Mitchell, T. M. (1977). Version spaces: A candidate elimination approach to rule learning. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (pp. 305–310). Cambridge, MA: MIT Press.

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

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

Mitchell, T., Mahadevan, S., & Steinberg, L. I. (1985). In *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 573–580). Los Angeles, CA: Morgan Kaufmann.

Moody, J., & Darken, C. (1988). Learning with localized receptive fields. In *Proceedings of the 1988 Connectionist Models Summer School* (pp. 133–143). Pittsburgh, PA: Morgan Kaufmann.

Mooney, R. J. (1987). *A general explanation-based learning mechanism and its application to narrative understanding* (Technical Report UILU-ENG-87-2269). Urbana, IL: University of Illinois, College of Engineering, Coordinated Science Laboratory.

Moore, A. W. (1990). Acquisition of dynamic control knowledge for a robotic manipulator. In *Proceedings of the Seventh International Conference on Machine Learning* (pp. 244–252). Austin, TX: Morgan Kaufmann.

Murphy, G. L., & Medin, D. L. (1985). The role of theories in conceptual coherence. *Psychological Review*, *92*, 289–315.

Neumann, P. G. (1974). An attribute frequency model for the abstraction of prototypes. *Memory & Cognition*, *2*, 241–248.

Newell, A. (1973). Artificial intelligence and the concept of mind. In R. C. Schank & K. M. Colby (Eds.), *Computer models of thought and language*. San Francisco, CA: Freeman.

Newell, A., Shaw, J. C., & Simon, H. A. (1963). Empirical explorations of the Logic Theory Machine: A case study in heuristics. In E. Feigenbaum & J. Feldman (Eds.), *Computers and thought.* New York, NY: McGraw-Hill.

Newell, A., & Simon, H. A. (1972). *Human problem solving.* Englewood Cliffs, NJ: Prentice-Hall.

Niblett, T., & Bratko, I. (1986). Learning decision rules in noisy domains. In *Proceedings of Expert Systems 1986* (pp. 25–34). Cambridge, England: Cambridge University Press.

Nosofsky, R. M. (1984). Choice, similarity, and the context theory of classification. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 10*, 104–114.

Nosofsky, R. M. (1986). Attention, similarity, and the identification-categorization relationship. *Journal of Experimental Psychology: General, 15*, 39–57.

Nosofsky, R. M. (1987). Attention and learning processes in the identification and categorization of integral stimuli. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 13*, 87–108.

Nosofsky, R. M. (1989). Further tests of an exemplar-similarity approach to relating identification and categorization. *Perception & Psychophysics, 45*, 279–290.

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.

Ortony, A., Vondruska, R. J., Foss, M A., & Jones, L. E. (1985). Salience, similes, and asymmetry of similarity. *Journal of Memory and Language, 24*, 569–594.

Overmars, M. H., & van Leeuwen, J. (1982). Dynamic multi-dimensional data structures based on quad- and k-d trees. *Acta Informatica, 17*, 276–285.

Pagallo, G. (1989). Learning DNF by decision trees. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 639–644). Detroit, MI: Morgan Kaufmann.

Pazzani, M. J. (1989). *Learning causal relationships by integrating empirical and explanation-based learning methods.* Hillsdale, NJ: Lawrence Erlbaum.

Pazzani, M. J., & Sarrett, W. E. (1990). Integrating empirical and explanation-based learning: Experimental and analytical results. In *Proceedings of the Seventh International Conference On Machine Learning* (pp. 339–347). Austin, TX: Morgan Kaufmann.

Penrod, C. S., & Wagner, T. J. (1977). Another look at the edited nearest neighbor rule. *Institute of Electrical and Electronic Engineers Transactions on Systems, Man and Cybernetics*, *7*, 92–94.

Pitt, L., & Valiant, L. G. (1986). *Computational limitations on learning from examples* (Technical Report TR-05-86). Cambridge, MA: Harvard University, Aiken Computation Laboratory.

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

Porter, B. W., & Bareiss, E. R. (1986). PROTOS: An experiment in knowledge acquisition for heuristic classification tasks. In *Proceedings of the International Meeting on Advances in Machine Learning* (pp. 159–174). Les Arcs, France.

Posner, M. I., & Keele, S. W. (1968). On the genesis of abstract ideas. *Journal of Experimental Psychology*, *77*, 353-363.

Posner, M. I., & Keele, S. W. (1970). Retention of abstract ideas. *Journal of Experimental Psychology*, *83*, 304–308.

Qian, N., & Sejnowski, T. (1988). Predicting the secondary structure of globular proteins using neural network models. *Journal of Molecular Biology*, *202*, 865–884.

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

Quinlan, J. R. (1986b). The effect of noise on concept learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. II). San Mateo, CA: Morgan Kaufmann.

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

Quinlan, J. R. (1988). An empirical comparison of genetic and decision-tree classifiers. In *Proceedings of the Fifth International Conference on Machine Learning* (pp. 135–141). Ann Arbor, MI: Morgan Kaufmann.

Quinlan, J. R. (1989). Unknown attribute values in induction. In *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 164–168). Ithaca, NY: Morgan Kaufmann.

Quinlan, J. R. (1990a). Probabilistic decision trees. In Y. Kodratoff & R. Michalski (Eds.), *Machine learning: An artificial intelligence approach* (Vol. III). San Mateo, CA: Morgan Kaufmann.

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

Quinlan, J. R., Compton, P. J., Horn, K. A., & Lazurus, L. (1986). Inductive knowledge acquisition: A case study. In *Proceedings of the Second Australian Conference on Applications of Expert Systems.* Sydney, Australia.

Reed, S. K. (1970). *Decision processes in pattern classification.* Doctoral dissertation, Department of Psychology, University of California, Los Angeles, CA.

Reed, S. K. (1972). Pattern recognition and categorization. *Cognitive Psychology, 3,* 382–407.

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

Rendell, L. (1988). Learning hard concepts. In *Proceedings of the Third European Working Session on Learning* (pp. 177–200). Glasgow, Scotland: Pitman.

Ringle, M. (1979). Philosophy and artificial intelligence. In M. Ringle (Ed.), *Philosophical perspectives in artificial intelligence.* Atlantic Heights, NJ: Humanities Press.

Rissland, E. L., & Ashley, K. D. (1988). Credit assignment and the problem of competing factors in case-based reasoning. In *Proceedings of a Case-Based Reasoning Workshop* (pp. 327–344). Clearwater Beach, FL: Morgan Kaufmann.

Rissland, E. L., Kolodner, J., & Waltz, D. (1989). Case-based reasoning *from* DARPA: Machine learning program plan. In *Proceedings of a Case-Based Reasoning Workshop* (pp. 1–13). Pensacola Beach, FL: Morgan Kaufmann.

Rivest, R. (1987). Learning decision lists. *Machine Learning, 2,* 1–20.

Robbins, D., Barresi, J., Compton, P., Furst, A., Russo, M., & Smith, M.A. (1978). The genesis and use of exemplar vs. prototype knowledge in abstract category learning. *Memory & Cognition, 6,* 473–480.

Robinson, A. J. (1989) *Dynamic error propagation networks.* Doctoral dissertation, Department of Engineering, Cambridge University, Cambridge, England.

Rosch, E. (1978). Principles of categorization. In E. Rosch & B. B. Lloyd (Eds.), *Cognition and categorization.* Hillsdale, NJ: Lawrence Erlbaum.

Rosch, E., & Mervis, C. B. (1975). Family resemblances: Studies in the internal structure of categories. *Cognitive Psychology, 7,* 573–605.

Rosenblatt, F. (1962). *Principles of neurodynamics.* New York, NY: Spartan.

Ross, B. H. (1989). Some psychological results on case-based reasoning. In *Proceedings of a Case-Based Reasoning Workshop* (pp. 144–147). Pensacola Beach, FL: Morgan Kaufmann.

Ross, D. T. (1977). Structured Analysis: A language for communicating ideas. *Institute of Electrical and Electronic Engineers Transactions on Software Engineering, 3*, 16–34.

Roth, E. M., & Shoben, E. J. (1983). The effect of context on the structure of categories. *Cognitive Psychology, 15*, 346–378.

Rumelhart D. E., McClelland, J. L., & The PDP Research Group (Eds.), (1986). *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. 1). Cambridge, MA: MIT Press.

Salzberg, S. (1988). *Exemplar-based learning: Theory and implementation* (Technical Report TR-10-88). Cambridge, MA: Harvard University, Center for Research in Computing Technology.

Salzberg, S. L. (1990). *Learning with nested generalized exemplars.* Boston, MA: Kluwer Academic Publishers.

Salzberg, S., Delcher, A., Heath, D., & Kasif, S. (1990). *Learning with a helpful teacher: Preliminary draft* (Technical Report JHU-90/14). Baltimore, MD: Johns Hopkins University, Department of Computer Science.

Samet, H. (1990). *The design and analysis of spatial data structures.* Reading, MA: Addison-Wesley.

Schank, R. C., & Abelson, R. P. (1977). *Scripts, plans, goals, and understanding.* Hillsdale, NJ: Lawrence Erlbaum.

Schlimmer, J. C. (1987a). *Concept acquisition through representational adjustment.* Doctoral dissertation, Department of Information and Computer Science, University of California, Irvine, CA.

Schlimmer, J. C. (1987b). *1985 Auto Imports Database* [Machine-readable data file]. Irvine, CA: University of California, Department of Information and Computer Science, UCI repository of machine learning databases and domain theories, ml-repository@ics.uci.edu (Producer, Distributor).

Schlimmer, J. C., & Fisher, D. (1986). A case study of incremental concept induction. In *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 496–501). Philadelphia, PA: Morgan Kaufmann.

Schlimmer, J. C., & Granger, R. H., Jr. (1986). Beyond incremental processing: Tracking concept drift. In *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 502–507). Philadelphia, PA: Morgan Kaufmann.

Sebestyen, G. S. (1962). *Decision-making processes in pattern recognition.* New York, NY: The Macmillan Company.

Seidel, R. (1987). On the number of faces in higher-dimensional Voronoi diagrams. In *Proceedings of the Third Annual Symposium on Computational Geometry* (pp. 181–185). Waterloo, Ontario: Association for Computing Machinery.

Sejnowski, T. J., & Rosenberg, C. R. (1987). Parallel networks that learn to pronounce English text. *Complex Systems, 1*, 145–168.

Selfridge, O. G., Sutton, R. S., & Barto, A. G. (1985). Training and tracking in robotics. In *Proceedings of the Ninth International Conference in Artificial Intelligence* (pp. 670–672). Los Angeles, CA: Morgan Kaufmann.

Shackelford, G. G., & Volper, D. J. (1987). *Learning in the presence of noise.* Unpublished manuscript. University of California, Department of Information and Computer Science, Irvine.

Shamos, M. I., & Hoey, D. (1975). Closest point problems. In *Proceedings of the Sixteenth Annual Institute of Electrical and Electronic Engineers Symposium on the Foundations of Computer Science* (pp. 151–162). Institute of Electrical and Electronics Engineers Computer Society.

Shepard, B. (1983). An appraisal of a decision tree approach to image classification. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence* (pp. 473–475). Karlsruhe, West Germany: William Kaufmann.

Shepard, R. N. (1964). Attention and metric structure of the stimulus space. *Journal of Mathematical Psychology, 1*, 54–87.

Shepard, R. N. (1987). Toward a universal law of generalization for psychological science. *Science, 237*, 1317–1323.

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.* San Mateo, CA: Morgan Kaufmann.

Smith, E. E., & Medin, D. L. (1981). *Categories and concepts.* Cambridge, MA: Harvard University Press.

Specht, D. F. (1990). Probabilistic neural networks. *Neural Networks, 3*, 109–118.

Spiegel, M. R. (1988). *Schaum's outline of theory and problems of statistics.* New York, NY: McGraw-Hill.

Sproull, R. F. (in press). Refinements to nearest-neighbor searching in k-d trees. *Algorithmica.*

Stanfill, C. (1987). Memory-based reasoning applied to English pronunciation. In *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 577–581). Seattle, WA: Morgan Kaufmann.

Stanfill, C. (1988). Learning to read: A memory-based model. In *Proceedings of a Case-Based Reasoning Workshop* (pp. 402–413). Clearwater Beach, FL: Morgan Kaufmann.

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

Stanfill, C., & Waltz, D. (1988). The memory-based reasoning paradigm. In *Proceedings of a Case-Based Reasoning Workshop* (pp. 414–424). Clearwater Beach, FL: Morgan Kaufmann.

Stepp, R. E., III, & Michalski, R. S. (1986). Conceptual clustering: Inventing goal-oriented classifications of structured objects. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. II). San Mateo, CA: Morgan Kaufmann.

Sturt, E. (1981). Computerized construction in Fortran of a discriminant function for categorical data. *Applied Statistics, 30*, 213–222.

Tan, M., & Schlimmer, J. C. (1990). Two case studies in cost-sensitive concept acquisition. In *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 854–860). Boston, MA: American Association for Artificial Intelligence Press.

Thompson, K., & Langley, P. (1989). Incremental concept formation with composite objects. In *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 371–374). Ithaca, NY: Morgan Kaufmann.

Tomek, I. (1976a). A generalization of the k-NN rule. *Institute of Electrical and Electronics Engineers Transactions on Systems, Man, and Cybernetics, 6*, 121–126.

Tomek, I. (1976b). An experiment with the edited nearest neighbor rule. *Institute of Electrical and Electronics Engineers Transactions on Systems, Man, and Cybernetics, 6*, 448–452.

Towell, G. G., Shavlik, J. W., & Noordewier, M. O. (1990). Refinement of approximate domain theories by knowledge-based neural networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 861–866). Boston, MA: American Association for Artificial Intelligence Press.

Tversky, A. (1977). Features of similarity. *Psychological Review, 84*, 327–352.

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

Utgoff, P. E., & Brodley, C. E. (1990). An incremental method for finding multivariate splits for decision trees. In *Proceedings of the Seventh International Conference on Machine Learning* (pp. 58–65). Austin, TX: Morgan Kaufmann.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the Association for Computing Machinery, 27*, 1134–1142.

Valiant, L. G. (1985). Learning disjunctions of conjunctions. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 560–566). Los Angeles, CA: Morgan Kaufmann.

Van de Velde, W. (1990). Incremental induction of topologically minimal trees. In *Proceedings of the Seventh International Conference on Machine Learning* (pp. 66–74). Austin, TX: Morgan Kaufmann.

Vapnik, V. N., & Chervonenkis, A. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications, 16*, 264–280.

Voisin, J., & Devijver, P. A. (1987). An application of the Multiedit-Condensing technique to the reference selection problem in a print recognition system. *Pattern Recognition, 5*, 465–474.

Volper, D. J., & Hampson, S. E. (1987). Learning and using specific instances. *Biological Cybernetics, 57*, 57–71.

Volper, D. J., & Hampson, S. E. (1990). Quadratic function nodes: Use, structure, and training. *Neural Networks, 3*, 93–107.

Voronoi, G. (1908). Novelles applications des parametres continus à la theorie des formes quadratique, deuxième mémoire: recherches sur les paralléllo¡èdres primitifs. *Journal Reine U. Angew. Math, 1*, 198–287.

Waltz, D. (1990). Massively parallel AI. In *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 1117-1122). Boston, MA: American Association for Artificial Intelligence Press.

Wagner, T. J. (1973). Convergence of the edited nearest neighbor. *Institute of Electrical and Electronic Engineers Transactions on Information Theory, 19*, 696–697.

Wasserman, K. (1985). *Unifying representation and generalization: Understanding hierarchically structured objects.* Doctoral dissertation,
Department of Computer Science, Columbia University, New York, NY.

Watson, D. F. (1981). Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *Computer Journal, 24*, 167–172.

Weiss, S. M., & Kapouleas, I. (1989). An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 781–787). Detroit, MI: Morgan Kaufmann.

Wilson, D. (1972). Asymptotic properties of nearest neighbor rules using edited data. *Institute of Electrical and Electronic Engineers Transactions on Systems, Man and Cybernetics*, *2*, 408–421.

Wilson, S. W. (1987). Classifier systems and the animat problem. *Machine Learning*, *2*, 199–228.

Winston, P. H. (1975). Learning structural descriptions from examples. In P. H. Winston (Ed.), *The psychology of computer vision.* New York, NY: McGraw Hill.

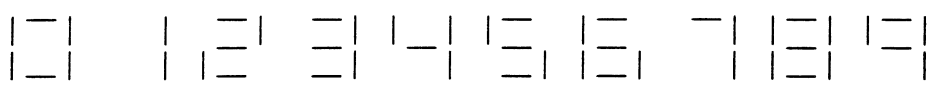
Wolpert, D. H. (1990). Constructing a generalizer superior to NETtalk via a mathematical theory of generalization. *Neural Networks*, *3*, 445–452.

Appendix A: Brief Descriptions of Applications

All of the databases and domains used in the symbolic and numeric prediction tasks that were used to evaluate the IBL algorithms in Chapters 4 and 5 can obtained from the U.C.I repository of machine learning databases and domain theories. Queries for information should be sent to ml-databases@ics.uci.edu. This section briefly describes these databases and data generators.

1. LED Display domains

Instances in the LED-7 domain are described by 7 binary-valued attributes that correspond to 7 light-emitting diodes. These settings correspond to one of the ten decimal digits, as shown below. This domain was corrupted with a 10% probability of attribute noise. That



is, each value of each instance's attribute was negated with a probability of 10%. The class probabilities are equal (i.e., 10% probability for each class). The Bayes optimal classification rate of this domain is 74%.

The LED-24 domain contains an additional 17 irrelevant binary-valued attributes whose values are randomly assigned. This domain was first investigated by Breiman *et al.* (1984, pp. 48).

2. Waveform domains

The Waveform-21 domain consists of 21 numeric-valued predictor attributes and three target concepts (Breiman *et al.*, 1984, pp. 49). Each class's instances are defined by a linear combination of two *waves*. Three waves are defined in terms of the 21 attributes using the function $f_c \in$ Attribute $\rightarrow \Re$ as follows:

$$f_1(i) = \begin{cases} i-1 & 2 \leq i \leq 7 \\ 5-(i-8) & 8 \leq i \leq 12 \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(i) = \begin{cases} i-9 & 10 \leq i \leq 15 \\ 5-(i-16) & 16 \leq i \leq 20 \\ 0 & \text{otherwise} \end{cases}$$

$$f_3(i) = \begin{cases} i-5 & 6 \leq i \leq 11 \\ 5-(i-12) & 12 \leq i \leq 16 \\ 0 & \text{otherwise} \end{cases}$$

Depending on which class the instance exemplifies, attributes x_i are assigned the following values:

$$\text{Class 1: } x_i = uf_1(i) + (1-u)f_2(i) + \epsilon_i$$
$$\text{Class 2: } x_i = uf_1(i) + (1-u)f_3(i) + \epsilon_i$$
$$\text{Class 3: } x_i = uf_2(i) + (1-u)f_3(i) + \epsilon_i$$

where u is a uniform random number and the random numbers ϵ_i are normally distributed with mean 0 and variance 1. The class probabilities are equal (i.e., $33.\bar{3}\%$ probability for each class). The Bayes optimal classification rate of this domain is approximately 86%. Note that the ϵ_i values are interpreted as attribute noise.

The Waveform-40 domain uses an additional 19 irrelevant attributes to describe each instance. Their values have a mean of 0 and a variance of 1.

3. Cleveland database

This domain was donated by Dr. Robert Detrano, M.D., and is described in (Detrano *et al.*, 1989). It consists of 303 patient diagnoses described by 13 predictor attributes collected at the Cleveland Clinic Foundation. The predictor attributes are:

1. Age in years
2. Gender, coded as 0 or 1
3. Chest pain type: typical (0), atypical (1), non-anginal (2), or asymptotic (3)
4. Resting blood pressure in mm Hg on admission to the hospital
5. Serum cholesterol in mg/dl
6. Fasting blood sugar: if above 120 mg/dl then 1 else 0
7. Resting ecg results: normal (0), abnormal (1), or indicative of hypertrophy (2)
8. Maximum heart rate achieved
9. Exercise-induced angina: if occurred then 1 else 0
10. ST depression induced by exercise relative to rest
11. The slope of the peak exercise ST segment, discretized as up (0), flat (1), or down (2)
12. Number of major vessels (0-3) colored by flourosopy
13. Type of defect: Normal (0), fixed (1), or reversible defect (2)

Four of these predictor attributes (i.e., numbers 3, 7, 11, and 13) are symbolic-valued but were interpreted as numeric-valued with the values shown. The class refers to the degree of heart disease diagnosed and ranges in $[0, 4]$. For symbolic prediction tasks, this value was predicted to be either 0 or positive. Three examples are:

```
52 1 2 120 325 0 0 172 0 0.2 1 0 3 fine
53 0 4 138 234 0 2 160 0 0.0 1 0 3 fine
56 1 4 125 249 1 2 144 1 1.2 2 1 3 sick
```

4. Hungarian database

This domain was also donated by Dr. Detrano. These 294 diagnoses were collected from the Hungarian Institute of Cardiology in Budapest, Hungary. The set of attributes are identical to the ones used in the Cleveland domain. However, the target attribute is restricted to binary values. Although the Cleveland database contains only 6 instances with missing values, this database contains 781 missing attribute values.

5. Congressional Voting database

The voting database consists of instances described by 16 binary-valued predictor attributes. The target attribute is political party, either democratic or republican. Each of the 335 instances describes the known (yes/no) voting record of a United States Congressional representative during the second session of 1984. The predictor attributes are

1. Handicapped infants
2. Water project cost sharing
3. Adoption of the budget resolution
4. Physician fee freeze
5. El Salvador aid
6. Religious groups in schools
7. Anti satellite test ban
8. Aid to Nicaraguan contras
9. Mx missile
10. Immigration
11. Synfuels corporation cutback
12. Education spending
13. Superfund right to sue
14. Crime
15. Duty free exports
16. South Africa export administration act

Although this 288 missing values appear in this database, Steve Hampson (personal communication) has shown that it is linearly separable along the political party target attribute.

6. Automobile database

This database, built from several sources by Jeff Schlimmer (1987b), contains 205 instances described by 25 predictor attributes (15 continuous and 10 nominal-valued). Each instance describes a 1985 imported automobile. I examined the car's horsepower rating in Chapters 4 and 5. The predictor attributes are:

1. Risk factor symbol (continuous)
2. Normalized-losses (continuous)
3. Make of automobile (22 types)
4. Fuel type (diesel or gas)
5. Aspiration (standard or turbo)
6. Number of doors (continuous)
7. Body style (5 types)
8. Type of drive (4 wheel, rear, front)
9. Engine location (front or rear)
10. Wheel-base (continuous)
11. Length (continuous)
12. Width (continuous)
13. Height (continuous)
14. Curb weight (continuous)
15. Engine type (7 types)
16. Number of cylinders (continuous)
17. Engine size (continuous)
18. Type of fuel system (8 types)
19. Bore (continuous)
20. Stroke (continuous)
21. Compression ratio (continuous)
22. Peak revolutions per minute (continuous)
23. City miles per gallon (continuous)
24. Highway miles per gallon (continuous)
25. Price (continuous)

A total of 59 values are missing in this database of 205 instances. Three example instances are:

```
 0  78  honda  std four wagon fwd  96.5 157.1 63.9 58.3 2024 ohc 4
92 1bbl 2.92 3.41  9.20  6000 30 34  7295 76hp

 0 118  mazda  std four sedan rwd 104.9 175.0 66.1 54.4 2670 ohc 4
140 mpfi 3.76 3.16  8.00 5000 19 27 18280 120hp

-1  74  volvo turb four wagon rwd 104.3 188.8 67.2 57.5 3157 ohc 4
130 mpfi 3.62 3.15  7.50 5100 17 22 18950 162hp
```

7. Echocardiogram database

This database was collected by Dr. Evlin Kinney of the The Reed Institute, located in Miami, Florida. It contains 132 diagnoses of patients who have suffered heart attacks. These instances are described by 9 numeric-valued predictor attributes:

1. Still alive: 0=dead at end of survival period, 1 means still alive
2. Age at heart attack: Age in years when heart attack occurred
3. Pericardial effusion: 0 if no fluid around the heart, otherwise 1
4. Fractional shortening: A measure of contractability around the heart
5. EPSS: E-point septal separation, another measure of contractability
6. Left ventricular end-diastolic dimension: A a measure of the size of the heart at end-diastole.
7. Wall motion score: A measure of how the segments of the left ventricle are moving
8. Wall motion index: The wall motion score divided by number of segments seen. Usually 12-13 segments are seen in an echocardiogram.
9. Alive at 1: Whether the patient lived after 1 year or had been followed for less than 1 year.

The target attribute used in this dissertation was the number of months patient survived since their heart attack. Three typical examples are

```
0 71 0 0.26 9  4.600 14 1    0 11
1 57 0 0.16 22 5.750 18 2.25 0 19
0 72 0 0.38 6  4.100 14 1.7  0 19
```

8. Breast cancer database

This database was donated by the Institute of Oncology at the University Medical Centre in Ljubljana, Yugoslavia. It contains 286 diagnoses of breast cancer patients. Nine predictor attributes were used and my experiments focused on predicting the size of the breast tumor detected. The predictor attributes included

1. Recurrence events: 1 if recurrence occurred, otherwise 0
2. Age
3. Time of menopause: before age 40, after 40, or pre-menopause
4. Number of nodes affected as observed by X-rays
5. Node caps (yes,no)
6. Degree of malignancy (1, 2, or 3)
7. Breast (left,right)
8. Breast quadrant (left-up,left-low,right-up,right-low,central)
9. Using irradiation treatment (yes,no)

Only 9 values are missing among this database's instances. Three example instances from this database are:

```
0 40 pre-menopause   0 0 2 1 right_up 0 size=20
0 40 pre-menopause   0 0 2 0 left_low 0 size=20
0 60 geater_than_40  0 0 2 1 left_up  0 size=15
```

Appendix B: Details of the Subject Experiments

This appendix details the results for the two subject experiments that were summarized in Section 6.6. The following details the percent subjects and GCM-ISW's percentage estimates for classifying each of the instances in the first experiment as a member of class 2:

Subject Averages:

	1	2	3	4	5	6	7	8
1	50	45	45	40	50	65	10	30
2	65	60	80	80	85	90	20	25
3	55	50	55	60	70	85	0	20
4	40	55	45	40	65	85	0	15
5	20	20	15	25	30	70	0	20
6	10	0	5	10	10	5	5	15
7	85	90	90	95	90	85	75	80
8	55	60	65	60	55	60	55	60

GCM-ISW Estimates:

	1	2	3	4	5	6	7	8
1	67	74	77	78	78	76	34	31
2	50	63	72	76	78	78	33	30
3	31	43	58	69	74	74	29	28
4	20	26	37	52	64	69	26	24
5	17	19	23	32	46	57	23	21
6	21	21	22	27	37	48	26	25
7	74	75	77	78	78	74	53	61
8	78	78	78	78	76	72	42	49

The subjects' results are the percentage of the subject who predicted that the instance was a member of class number 2. GCM-ISW's results reflect its estimate of the probability that the instance is a member of this same class. The correlation of these results is 0.78. The two sets of results agree on 50 of the 64 instances' classifications, but they disagree on most classification in the top-left and bottom-right corners of the instance space. That is, one result favors class 1 (i.e., below 50%) while the other favors class 2 (i.e., at least 50%).These also happen to be the areas that are located furthest from the given data. In most of these disagreements, GCM-ISW favored class 1 while the subjects favored class 2. These majority subjects' classifications and favored classification by GCM-ISW in the first experiment are:

Subject's Classifications:

	1	2	3	4	5	6	7	8
1	2	1	1	1	2	2	1	1
2	2	2	2	2	2	2	1	1
3	2	2	2	2	2	2	1	1
4	1	2	1	1	2	2	1	1
5	1	1	1	1	1	2	1	1
6	1	1	1	1	1	1	1	1
7	2	2	2	2	2	2	2	2
8	2	2	2	2	2	2	2	2

GCM-ISW's Classifications:

	1	2	3	4	5	6	7	8
1	2	2	2	2	2	2	1	1
2	1	2	2	2	2	2	1	1
3	1	1	2	2	2	2	1	1
4	1	1	1	2	2	2	1	1
5	1	1	1	1	1	2	1	1
6	1	1	1	1	1	1	1	1
7	2	2	2	2	2	2	2	2
8	2	2	2	2	2	2	1	1

It is not obvious why these particular locations (i.e., top-left, lower-right instances, and main diagonal) are the ones that differ between the two sets of predictions. However, these are

the instances that are most in doubt amongst all 64 instances in the space. GCM-ISW's correlation on the four instances of special interest was extremely high (0.97), which shows that the subjects also used a something akin to a localized attribute weighting scheme in these areas.

GCM-NW and GCM-SW's estimates of the probability that the instances are classified as members of class 2 in the first experiment are:

GCM-NW's Estimates:

	1	2	3	4	5	6	7	8
1	63	68	72	75	77	75	69	62
2	55	61	67	71	74	77	62	52
3	43	49	57	63	66	65	41	39
4	32	36	42	50	58	57	34	32
5	26	27	31	37	43	40	26	28
6	27	24	30	36	47	47	39	34
7	42	50	65	70	73	66	57	49
8	55	62	69	73	75	72	67	61

GCM-SW's Estimates:

	1	2	3	4	5	6	7	8
1	70	73	75	77	78	77	73	67
2	62	66	70	73	75	78	66	57
3	50	55	60	63	65	64	42	41
4	35	39	45	50	54	55	35	33
5	26	27	30	34	38	35	25	27
6	28	24	28	33	42	44	40	36
7	47	55	67	72	74	68	61	55
8	61	67	72	75	76	75	71	67

Their correlations with the subject data was 0.65 and 0.66 respectively. GCM-NW's results are relatively difficult to analyze, but it is obviously not fitting the subject data as well as GCM-ISW. GCM-NW guesses wrong on three of the four instances of special interest and its correlations with the subjects on these instances is poor (i.e., -0.42). GCM-MW's results are a bit easier to analyze. It tends to favor giving the same classifications to all instances in the same row, which applies for all but rows three and four. This occurred because GCM-MW assigned a higher weight to the x axis than the y axis. This causes predictions to be more influenced by the predictions for instances along the same row than for instances on different columns. GCM-MW's low correlation with the subjects on the 4 special instances (i.e., 0.17) suggests that the subjects do not classify these instances using a strategy similar to GCM-MW's.

The results for the subjects and GCM-ISW for the second experiment are:

Subject's Averages:

	1	2	3	4	5	6	7	8
1	65	50	65	55	55	60	65	60
2	70	75	85	90	90	90	90	80
3	15	10	15	15	10	10	0	10
4	15	5	85	35	15	15	5	25
5	20	5	90	60	35	40	50	35
6	20	0	85	70	50	60	50	60
7	30	15	90	80	75	75	60	60
8	30	10	70	55	55	50	55	50

GCM-ISW's Estimates:

	1	2	3	4	5	6	7	8
1	47	39	69	75	77	78	78	78
2	59	50	72	77	77	77	76	74
3	24	25	46	36	27	23	21	21
4	22	22	58	49	36	28	23	21
5	25	26	68	65	55	42	32	25
6	28	30	75	74	70	61	48	36
7	30	31	76	77	76	72	65	54
8	29	31	74	77	77	76	74	68

GCM-ISW's correlation with the subject data for this experiment was quite good (i.e., 0.86). Its correlation on the four special instances was 0.95. Its poorest fits were again located in regions of greatest doubt – near the corners intersected by the main diagonal. GCM-NW and GCM-MW's results for the second experiment are:

GCM-NW's Estimates:

	1	2	3	4	5	6	7	8
1	65	70	74	75	74	71	65	59
2	53	60	67	74	71	66	53	45
3	36	40	45	44	34	29	24	27
4	27	25	37	40	36	30	27	26
5	33	35	56	55	50	44	38	34
6	40	42	65	65	63	59	53	47
7	55	64	78	75	72	69	65	60
8	65	71	76	77	76	74	71	68

GCM-MW's Estimates:

	1	2	3	4	5	6	7	8
1	55	64	71	74	72	67	59	51
2	44	53	64	73	69	63	47	39
3	33	38	49	51	38	31	25	27
4	28	27	43	47	39	32	27	25
5	31	33	59	57	50	41	34	30
6	37	40	66	67	62	53	45	38
7	48	59	76	74	70	63	56	48
8	58	66	74	76	74	69	63	57

Their overall correlations were 0.68 and 0.67 respectively and -0.84 and -0.42 for the four instances of special interest in the second experiment. In summary, these results strongly suggest that GCM-ISW's localized weighting strategy is more similar to the strategy used by the subjects than the strategies used by either GCM-NW or GCM-MW.