Naval Research Laboratory

Washington, DC 20375-5000

DTIC

AD-A242 472

‖‖‖‖‖‖‖‖‖‖‖‖‖‖‖‖‖

Machine Learning Systems:
Part I — Concept Learning from Examples with
AQ15 and Related Systems

DIANA F. GORDON AND WILLIAM M. SPEARS

Naval Center for Applied Research in Artificial Intelligence
Information Technology Division

September 30, 1991

91-15199
‖‖‖‖‖‖‖‖‖‖‖‖‖‖‖‖‖‖‖

91 1106 04?

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 30, 1991	3. REPORT TYPE AND DATES COVERED Final 1989-1990

4. TITLE AND SUBTITLE Concept Learning Systems: Part I — Concept Learning from Examples with AQ15 and Related Systems	5. FUNDING NUMBERS PE — 62234N TA — RS34-C74-000
6. AUTHOR(S) Diana F. Gordon and William M. Spears	

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Research Laboratory Washington, DC 20375-5000	8. PERFORMING ORGANIZATION REPORT NUMBER NRL Report 9330

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This is the first in a series of reports designed to acquaint Navy and other military personnel with current software available for machine learning. By acquainting personnel with the available software, we encourage the applicability of learning systems. AQ15 is a concept learning system that has the advantages of user-definable parameters and easily readable output.

14. SUBJECT TERMS Concept learning Induction Machine learning	15. NUMBER OF PAGES 29
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

CONTENTS

MACHINE LEARNING SYSTEMS:
PART I — CONCEPT LEARNING FROM EXAMPLES WITH AQ15 AND RELATED SYSTEMS

INTRODUCTION

As technology rapidly advances, our reliance on computers to alleviate some of the computational burden increases. The Navy has experts in many diverse areas including warfare tactics, electronics diagnosis, and signal processing. The introduction of *expert systems* technology has, for the first time, brought artificial intelligence to a very practical level and has encouraged experts in many areas to consider the advantages of computers as assistants. Prompted by the increase in systems' complexity and the cost of maintaining Navy systems, the Navy has established a Center for Applied Research in Artificial Intelligence (NCARAI) at the Naval Research Laboratory. The mission of this Center is to explore and develop research in artificial intelligence and then to transition the results into products useful to the Navy. One of the Center's first and most successful projects is the Fault Isolation System, an expert system used for trouble-shooting electronic equipment (Pipitone et al. 1988). There are many other areas where expert systems could be equally useful for the Navy. However, there is another area of interest at NCARAI besides expert systems that is equally promising for the Navy. This is the area of machine learning. Machine learning has two major goals. The first is to program machines to alleviate the costly process of transferring and summarizing knowledge obtained from experts. The second goal is to program machines to acquire and organize knowledge in a useful way in domains for which no experts exist. Machine learning has tremendous potential for increasing the Navy's monetary savings if these goals are satisfied.

One of the tasks of NCARAI is to transition basic artificial intelligence research into applications of interest to the Navy. To satisfy this goal, the Machine Learning Group at NCARAI has started the Systems Collection Project. This project consists of collecting, studying, documenting, and disseminating information about machine learning systems with the eventual goal of applying these systems to Navy problems.* Several steps are involved in the task of transitioning machine learning software from research to Navy applications. The first step is acquiring each system. There is then a process of becoming more familiar with the strengths and weaknesses of the system by studying existing documentation by the author(s). After reading the literature, it is important to learn how to use the system and execute it with different data sets to be able to characterize its performance. Next, information about the system must be dispersed to Navy employees who are not computer scientists but who might benefit from machine learning technology. The final step is to apply the system to Navy problems whenever suitable. We have completed all except the last two steps for several systems. This report is the first in a series of reports designed to disseminate information about the machine learning systems that have been collected at NCARAI.

*The remainder of this report uses the term "system" synonymously with "computer program."

In this report, we discuss AQ15, a machine learning system that induces general concepts (i.e., rules for classifying new facts) from specific examples of the concept, as well as a predecessor (NEWGEM) and a successor (POSEIDON) of AQ15 (Hong et al. 1986; Mozetic 1985; Bergadano et al. forthcoming).* We begin by introducing the machine learning subfield called *concept learning*— the process of learning (inducing) general concepts from specific examples of the concept. Since it is frequently more difficult for experts to verbalize the expert rules they use than to produce specific examples of the rules, systems (like AQ15) that automate the process of learning from examples have potential advantages for the Navy.

Here we present categories of concept learning and explain where AQ15 fits within these categories. We depict concept learning as a search for the desired concept that is constrained by a user-defined set of criteria. We present AQ15's learning algorithm, describe features of AQ15 for handling realistic data, and present previously published results. Finally, we present new experimental results of running AQ15 on both natural and artificially constructed data sets. The purpose of this report is to introduce the capabilities and limitations of AQ15 and to allow personnel in diverse areas of Navy applications to understand and evaluate the potential for using such a program to help with their particular problems.

CONCEPT LEARNING

Machine learning is a subfield of artificial intelligence. Concept learning is a subfield of machine learning. There are two categories of concept learning—*knowledge acquisition* and *skill refinement*. Knowledge acquisition consists of acquiring new knowledge. Researchers who build systems for knowledge acquisition hope to bypass or alleviate the costly process of programming domain-specific (e.g., expert systems) expertise directly into the system. Skill refinement reformulates existing knowledge to improve a system's performance on a problem-solving task.

Here we focus on the knowledge acquisition category of concept learning. The process of acquiring concepts consists of inducing generalized descriptions from specific instances, or examples, of the concept. These generalized descriptions, also called *hypotheses*, are used to approximate the concept being learned (i.e., the *target concept*) as well as possible given the instances so far presented. Concept learning is often performed by *induction*, or *inductive inference*, which consists of extracting general rules, concepts, or other data structures from specific facts. Induction differs from deduction because although one can logically infer the facts from the generalization obtained via induction, one cannot, in general, deduce the generalization from the facts by using strict rules of logical inference. Therefore, the inference from the specific facts to the hypotheses, or generalizations, is not truth-preserving. The inference is, however, falsity-preserving. For example, suppose we have facts F and hypothesis H. If the inference used to derive H from F is deduction, then if F is true, H must necessarily be true. If the inference used to derive H from F is induction, then if F is true, H may or may not be true. However, if H is inductively inferred from F, then if H is true, F must be true. Furthermore, if some facts falsify F, then they must also falsify H, i.e., induction is falsity-preserving.

To illustrate concept learning, suppose one is trying to learn the concept "bird." If two examples of this concept have been encountered and both have wings, then birds might be described as "animals with wings." The goal of concept learning is to extract the important features (attributes) that describe all members of the concept.

---

*AQ15 and NEWGEM are written in Berkeley Pascal and run under the Unix operating system on VAX and SUN machines. The POSEIDON system is a recent development and has not yet been obtained for the NCARAI collection.

Concepts may be learned with or without a teacher. When concepts are learned without a teacher, the learning is called *descriptive generalization*. Examples of descriptive generalization include discovering patterns in observational data and formulating taxonomies to categorize objects. When some kind of teacher is involved in the learning process, the process is called *concept acquisition*. Here we focus on concept acquisition rather than descriptive generalization. The help teachers most frequently provide is that of classifying objects as positive or negative examples of the target concept. Suppose, for example, that the teacher is a human who is trying to teach the concept "bird." The teacher might present descriptions of birds and nonbirds. For the learning system, the teacher might be a human giving examples or it might be examples stored in a database based on experience (e.g., lists of symptoms of patients who do or do not have some disease). If the learning system is contained within some physical form (such as a robotic arm) that can interact with the environment, then the environment may play the role of the teacher. For instance, by applying pressure to objects, a robot can learn the concept of "breakable" objects.

Once examples have been classified, the next step in concept learning is for the system to take the descriptions of the specific examples provided and formulate a generalized description that fits the positive examples and does not fit the negative examples provided by the teacher. This generalized description, or hypothesis, is formed by taking those features that are common to the positive examples and that are not shared by negative examples. The purpose of a hypothesis is to approximate the target concept based on all examples encountered so far. If a hypothesis implies no known (previously seen) negative examples, then the hypothesis is considered to be *consistent* with the examples. Consistency can be achieved when all previous examples are saved. A hypothesis that implies all known (previously seen) positive examples is considered to be *complete*. For instance, the description of a bird as "an animal with wings" would be consistent and complete with respect to the positive example of an animal that is "small, brown, and has feathers and wings" and the negative example consisting of an animal that is "large, grey, mammalian, lives in the water, and has no feathers or wings."

AQ15 uses inductive inference to formulate hypotheses about a concept from positive and negative examples of the concept. AQ15 can learn multiple concepts. Instances are grouped according to the concept for which they are positive examples. AQ15 derives a hypothesis for each concept (class). This hypothesis is consistent and complete with respect to the given *unambiguous* examples of that class. An instance is unambiguous if it is an example of only one class. If an instance is an example of more than one class (i.e., the concepts are not disjoint), then it is considered *ambiguous*. Each concept is learned by treating the examples of that class as positive examples and the examples of all other classes as negative examples. Later in this report, we describe AQ15's method for handling ambiguous instances. Whenever there is a single target concept, AQ15 treats the negation of the target concept as an additional target concept. Therefore, in this case, the system maintains two hypotheses for learning two concepts. Both hypotheses are consistent and complete with respect to the instances that they are intended to cover.

AQ15'S METHOD FOR LEARNING CONCEPTS

The process of concept learning consists of a search through a space of possible descriptions to find the best description (hypothesis) that explains the instances (Michalski 1983). (See Nilsson 1980 for further information about search methods used in artificial intelligence.) Consistent and complete hypotheses are often preferred, though many other preference measures are used as well.

The ultimate goal of the learner is to induce a hypothesis that is identical to the target concept. If all examples are seen at once (i.e., learning is in *batch*), then it is possible to induce the correct concept in a single step. However, in many realistic concept learning situations, not all examples from which to learn are available at once. Often, instances are presented to the learner one or a few at a time. With the introduction of each new instance or subset of the instances, the learner must form a new hypothesis. When only a subset of the instances is introduced at a time, the learning is considered *incremental*. AQ15 can learn either in batch or incrementally.

The remainder of this section shows how AQ15's concept learning is a search through a space of possible descriptions. We first present the language in which any description of AQ15 may be expressed. We then present examples of generalization operators, which are used to form or modify hypotheses based on examples. This is followed by a description of the concept learning algorithm as well as the type of knowledge used to constrain the search. The section concludes with a description of AQ15's input and output requirements.

Instance and Hypothesis Language

When describing the language of the instances and hypotheses, we assume the reader is familiar with the basic terminology of formal logic. If not, an excellent introduction may be found in Hamilton 1978. AQ15 uses a version of first-order predicate calculus that has been modified to more easily express inductive generalizations. A generalization of a hypothesis is a modification to that hypothesis to make it match (imply) more instances. In AQ15, instances are described as a conjunction of *selectors*. A selector consists of a term, followed by a relational symbol ($<$, $>$, $<=$, $>=$, $=$, $<>$), followed by a value or a set of values. Selectors within instance descriptions have only a single value following the relational symbol "=." For example, an object might be described as "[size = 10] [color = blue] [shape = sphere]." The combination of the description of an instance and the classification of the instance as a positive or negative example of the target concept is called an *example* (or an *event*).

AQ15 takes one or more examples and from them, it formulates hypotheses. Hypotheses are in disjunctive normal form. Each disjunct (i.e., term or *complex*) consists of a conjunction of selectors. A disjunction of feature values within a selector, e.g., "[shape = sphere v cube]," called *internal disjunction*, is also allowed within hypotheses. An internal disjunct consists of a group of alternative values for a feature. For features with numeric domains, the range of alternative values is expressed by its endpoints. An example complex might be "[size = 1..50] [shape = sphere v cube]." A hypothesis is a disjunction of complexes, i.e., hypotheses are in a variant of disjunctive normal form (DNF) that allows internal disjunction.[*] A hypothesis is *satisfied* (i.e., consistent with all instances seen so far) if any of its complexes are satisfied, while a complex is satisfied if all selectors in it are satisfied (Hong et al. 1986). A hypothesis *covers* a set of instances if it implies the description of each instance in the set. A hypothesis that covers a set of instances may be sufficiently general to cover instances not yet seen as well. The satisfaction of a hypothesis is a measure of its consistency whereas the coverage of a hypothesis is a measure of its completeness with respect to a given set of instances.

The type of domains AQ15 can handle for its features (attributes) are *nominal, numeric,* and *structured nominal*. A nominal domain is unordered. An example of a nominal domain is the set of values {spoon, knife, fork} for the feature "eating utensil." AQ15's numeric domains consist of

---

[*] Although what we here call "hypotheses" are called "covers" in Michalski 1983, Reinke and Michalski 1988, Michalski et al. 1986, and Hong et al. 1986, we change the terminology here to avoid confusion between the noun and the verb forms of the term "cover."

integers or intervals of integers. A structured nominal domain has extra values in addition to the feature values present in the instances (which are nominals). These extra values (which are also nominals) are values to which a system may generalize. The set of all (including the extra) values of a structured nominal feature may be ordered by their degree of generality in a *generalization tree*, such as the ones shown in Fig. 1. Every structured nominal feature has one associated generalization tree. Let us illustrate by using the generalization tree for the feature "material" shown in Fig. 1. If values of the feature "material" that appear in the instances are "copper" and "aluminum," then these two feature values are specific cases of the more general value "element." "Copper" and "aluminum" may be considered *children* values of the *parent* value "element." A parent value is always more general than its children values. The *root* of a generalization tree (i.e., the value at the top of the tree) implies every value of the feature. Roots are labeled with the name of the feature. The *leaves* of a generalization tree (i.e., the values at the bottom of the tree) are the feature values present in the instances.
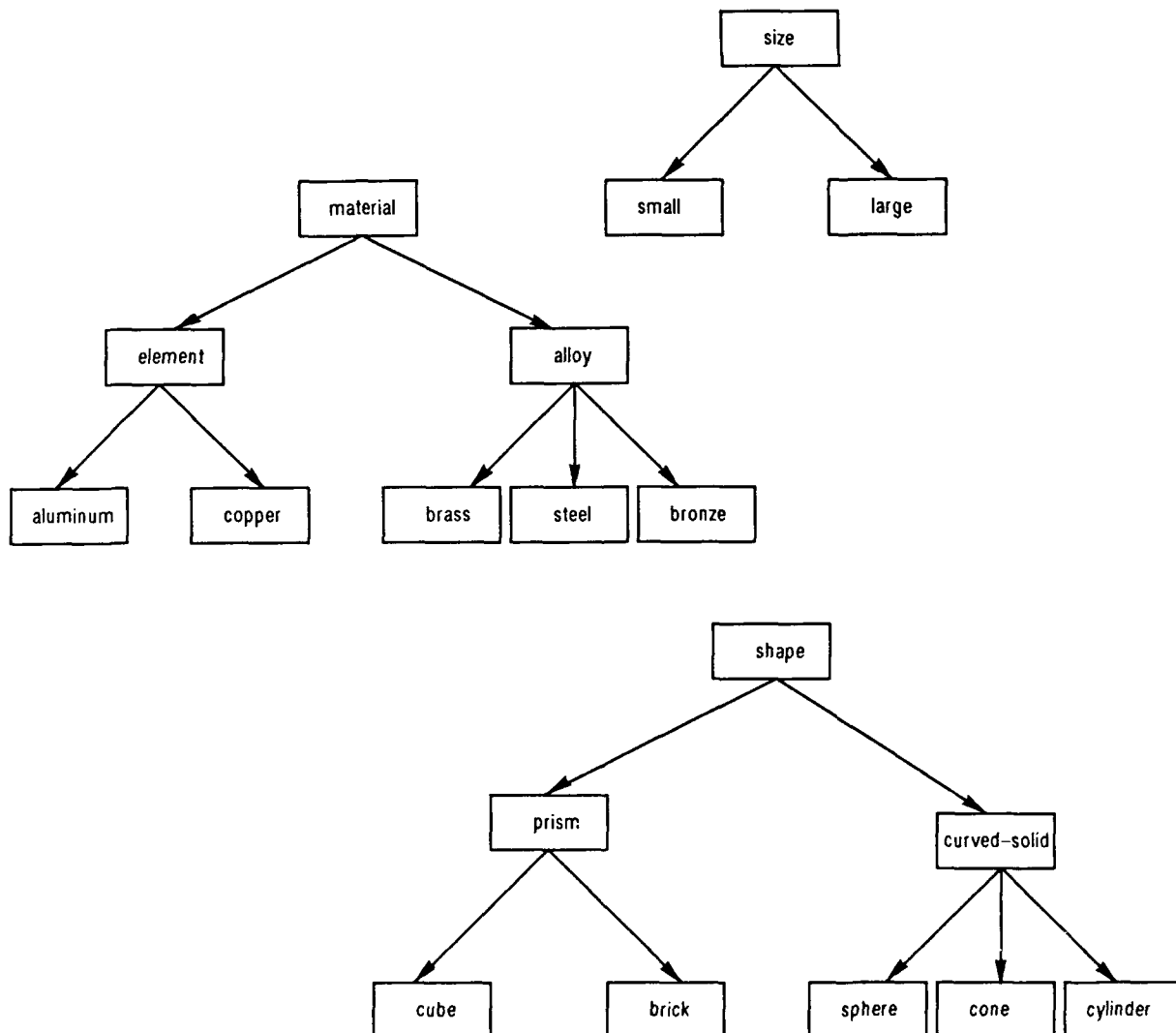

Fig. 1 — Generalization trees of descriptive terms

Generalization Operators

Now that we have described the language of the instances and hypotheses, we next present AQ15's generalization operators and rule formats for applying background knowledge. Generalization operators are used to form hypotheses from the instances. The operators are divided into two categories, those for selective induction and those for constructive induction. The difference between selective and constructive induction operators is that constructive induction operators generate new features to add to the hypothesis language, whereas selective induction operators do not. Constructive induction typically uses background knowledge about the domain.

AQ15 uses a number of selective induction operators for generalization. One very simple operator drops a feature when creating a hypothesis. For example, suppose the description of the first instance presented to AQ15 is "[size = 10] [color = blue] [shape = brick]," and then the second instance is described by "[size = 10] [color = blue] [shape = cube]." Furthermore, suppose these two instances are positive examples of the concept. A generalization that satisfies (is consistent with) both these instances is "[size = 10] [color = blue]." This hypothesis is formed by dropping the feature "shape" from either instance description. This hypothesis is more general than the disjunction of the two instance descriptions because it also describes blue objects of size 10 that have shapes other than brick or cube, e.g., spheres. It is now easy to see why induction is not truth-preserving. To illustrate, suppose the system uses its hypothesis to predict that blue objects of size 10 are going to be positive examples. The system might encounter a blue sphere of size 10 that is a negative example. This is called a *generalization error* and must be remedied. AQ15's method (*specialization*) for handling such errors will be discussed in the section on realistic data.

Another AQ15 generalization operator can form a hypothesis by adding an internal disjunct. For example, a new hypothesis for the two instances in the last paragraph could be "[size = 10] [color = blue] [shape = brick v cube]." If there is a generalization tree of terms (i.e., the feature is of type structured nominal), then AQ15 has yet another applicable generalization operator. This operator replaces the specific terms with the more general term. If the two instances presented in the preceding paragraph are used along with the generalization trees of Fig. 1, then this generalization operator would produce the hypothesis "[size = 10] [color = blue] [shape = prism]."*

The last selective induction operator used by AQ15 that we describe here is for numeric features. If two positive examples given to AQ15 are described by "[size = 10] [color = blue] [shape = brick]" and "[size = 20] [color = blue] [shape = brick]," then AQ15 can use this operator to generalize to a numeric interval, e.g., form the hypothesis "[size = 10..20] [color = blue] [shape = brick]." We next describe AQ15's rule formats for applying background knowledge.

Constructive induction operators derive new features (descriptors) not present in the original hypothesis language. The constructive induction operator implemented in AQ15 is called an *A-rule* (Hong et al. 1986). Arithmetic rules (A-rules) consist of arithmetic formulas for deriving new terms. An example of an A-rule is the following:

IF [length = L] and [width = W] THEN [area = L x W].

This rule is used to replace references to length and width (which were present in descriptions of the instances) with references to area in the hypotheses.

---

*AQ15 actually applies this operator during the postprocessing phase of learning, which is described in the following section

A type of AQ15 rule that is similar to the A-rule is the *L-rule* (logical assertion) (Hong et al. 1986). L-rules do not derive new features but are nevertheless similar to A-rules because they use background knowledge for their transformations. L-rules are frequently used to determine the value of one feature based on knowledge of other features. An example of an L-rule, or logical assertion, would be the rule:

IF [shape = cube] THEN [size = 10..20].

Star Algorithm

AQ15 adopts a generate-and-test approach for performing induction. Alternative hypotheses are first generated by the system and then tested according to a user-specified set of criteria. We begin by describing AQ15's Star algorithm for concept learning, which is the system's method for generating hypotheses. In the following section, we describe AQ15's criteria for selecting preferable hypotheses.

At the heart of AQ15 is the Star algorithm for inducing concepts from positive and negative examples of the concept. The algorithm builds each complex (disjunct) of a hypothesis from a *seed*, which is a randomly-chosen positive example. In essence, the algorithm continually generates general but consistent complexes from seeds and forms a hypothesis from the disjunction of these complexes. New complexes are added to the hypothesis until all the positive examples are covered (i.e., the hypothesis is complete). This is repeated for each concept. The Star algorithm is as follows (Hong et al. 1986; Michalski et al. 1986):

STEP 1:
Input the data.
STEP 2:
Preprocessing phase. Use background knowledge to generate new features and to select the most relevant features.
STEP 3:
Select a concept to learn. Acquire a labeling (from a teacher) of the instances as positive and negative examples of that concept.
STEP 4:
If incremental learning feature used, then an additional portion is added here as described in the section of this report on handling realistic data. If not, then continue to STEP 5.
STEP 5:
Select an uncovered positive example, i.e., a seed.
STEP 6:
Generate a *star*. A star is the set of maximally general complexes covering the seed and no negative examples.
STEP 7:
select a single, best complex (disjunct) from the star according to the *lexicographic evaluation function* (LEF) criteria (see below). Add the best complex to the current hypothesis.
STEP 8:
If all positive examples are now covered, then go to STEP 9. Otherwise, loop back to STEP 5.
STEP 9:
Postprocessing phase. Perform transformations of the hypotheses according to user-specified parameters.
STEP 10:
If done for all concepts, then end of algorithm. Otherwise, loop back to STEP 3.

The above algorithm begins with an initial hypothesis that is either empty, has previously been learned, or is supplied by the user. The portion of the algorithm where the star is generated—namely, STEP 6—is described in Michalski et al. 1986. This algorithm iteratively selects a seed, produces a complex that covers the seed and excludes negative examples, then adds the new complex to the current hypothesis. We have now described the generate portion of AQ15's generate-and-test method. Next we define AQ15's LEF, which is the test portion of its generate-and-test method.

Search Constraints: LEF and Other System Parameters

In addition to using background knowledge for constructive induction rules, AQ15 also uses background knowledge to constrain the inductive search space. Each feature of AQ15 has a domain-specific *annotation* consisting of relevant background knowledge (Michalski 1983) that guides the generalization process. For example, if a feature is annotated with its domain type, (e.g., numeric), then only generalization operators relevant for numeric features will be applied. Annotations, A-rules, and L-rules may all be used in the preprocessing phase (STEP 2) to generate new features and to select the most relevant features from the input set.

AQ15 has another method for constraining its inductive search. This important method consists of the application of an evaluation function for selecting the "best" hypothesis when there are multiple hypothesis from which to choose. Most scientists are familiar with the phenomenon that it is possible to form more than one (in fact, often many) hypothesis that all seem to explain the data. When one hypothesis is chosen from many, the criteria used for selecting are called the *biases* (Mitchell 1980). Examples of biases are hypothesis simplicity, degree of fit of the hypothesis to the data, consistency, and completeness. The consistency and completeness biases are built into AQ15's concept learning algorithm (the Star algorithm) described above. The simplicity and degree of fit biases, on the other hand, are part of the criteria used to select hypotheses.

AQ15's criteria for selecting hypotheses to bias induction is called the *lexicographic evaluation function* (LEF). Only a user-specified number (held in parameter *maxstar*) of complexes are kept in AQ15 at any time. The criteria used for selecting the "best" complexes (and therefore the "best" hypotheses, which are constructed from complexes) is the LEF. The LEF consists of the following criteria:

1. Maximize the number of newly-covered examples.

2. Minimize the number of features.

3. Minimize the (user-given) cost of features.

4. Minimize the total number of examples covered.

5. Maximize the number of features.

6. Maximize the ratio between newly covered positive examples and ʌ" covered negative examples.

7. Maximize the ratio between the total number of positive and negative examples covered.*

8. Minimize the total number of selectors in a complex.

---

*Although the hypotheses are consistent, Criteria 5 and 6 are part of the LEF because the LEF may also be used (during STEP 6 of the Star algorithm) for ordering partially formed complexes that might cover negative examples. See Michalski 1983 for details.

It is the job of the AQ15 user to order the LEF criteria according to importance. A subset of the above list is selected and ordered by the user prior to program execution. After each criterion, the user specifies a tolerance, which yields information about how strictly the criterion must be obeyed. By allowing the user to rate the criteria, the LEF approach offers an advantage to researchers who would like a degree of control over the induction process. For example, by rating criterion 5 as high, the user can force AQ15 to form very specific, rather than general, hypotheses. On the other hand, if the simplicity (measured in terms of length of the expression) of concepts is critical for the user, then the user could rate criterion 2 as highest priority. Simplicity may be desirable if the user wishes to verify AQ15's result because simplicity often results in increased readability.

The disadvantage of the LEF approach is that the quality of AQ15's results is sensitive to the setting of the LEF priorities. If the user does not wish to become involved enough with the details of AQ15 and the application to determine a good order for the LEF criteria, then the system will probably produce less than ideal results. It is possible to avoid having to select any of the system parameters by simply using the defaults, which are designed for good overall performance but are not optimized for any particular database.

Other important user-specified parameter settings, besides the LEF, include the *trim* and *maxstar*. Trim is used during the postprocessing phase (STEP 9) of the Star algorithm. It specifies the form of the hypotheses. The choice of values for this parameter are: "gen," which will favor more general hypotheses, "mini," which prefers simpler hypotheses, and "spec," which favors hypotheses that are as specific as possible (i.e., minimally general). It is often the case that the simplest hypothesis is also the most general, but this is not always true. A counterexample is the pair of hypotheses: "red" and "red or yellow or blue or green." The former is more simple and the latter is more general.

Maxstar is another user-specified parameter. Maxstar, which is is used in STEP 6 of the Star algorithm, controls the number of alternative solutions kept during complex formation. A higher number may improve solution quality but will be more computationally expensive. Other parameter settings are described in Mozetic 1985.

Program Input and Output

AQ15 requires five major items as program input. The first required item is a list of parameter (e.g., trim, maxstar) settings. This list typically consists of approximately seven or eight parameters and their settings. If the LEF criteria is not the default, then the LEF must also be specified. The second required input item is a list of the names of the attributes (features of the instances). AQ15 can handle a maximum of 60 attributes. The third item to be specified is the number of values for each attribute (including numeric attributes), which cannot exceed 58.* Fourth, for all structured nominal features, the information necessary for the system to construct a generalization tree must be entered as input. Finally, every instance must be presented as input to the system. The documentation does not specify a limit on the number of instances. Each instance is described by its attribute values. Values for integer features may be either integers or integer intervals. Instances must be grouped according to the class of which they are a member. For example, if AQ15 is learning the concepts "bird" and "dog," then all examples of the concept "bird" would be grouped together and

---

* The system can be reconfigured by a programmer to handle more attributes and values.

all examples of the concept "dog" would be in a separate group. Other inputs to the AQ15 system, besides these five, are optional and are described in Mozetic 1985. AQ15's format for inputting da ⅃ is reasonable. Nevertheless, some of the entries require redundant information.

AQ15's output consists of a hypothesis for each class that the system learns. These hypotheses are expressed in DNF. Each disjunct of each hypothesis has two numbers associated with it. The first is the number of instances that this disjunct covers; the second is the number of instances uniquely covered by this disjunct. AQ15's output format is very readable, probably because it is so similar to the way humans express concepts.

ABILITY TO HANDLE REALISTIC DATA

This section describes additional features of AQ15 and related systems that may be advantageous for applying the system within certain domains. We first present an optional feature offered by AQ15, namely, its incremental learning capability. We then discuss AQ15's method for handling noisy data. The system's ability to handle incomplete data is then briefly discussed. Finally, we discuss the flexible concept capability that is present in POSEIDON, a successor of AQ15.

Earlier we defined batch learning as learning a concept after all instances have been presented. Incremental learning is defined as inducing hypotheses to approximate a concept after each subset of the instances has been introduced. AQ15 is capable of learning in either of these two modes. Certainly, most everyday human learning is incremental. We rarely have the opportunity to learn with all the data present at the outset. The best we can do is to hypothesize based on the data so far encountered. For some realistic applications, it is essential that a learning system be able to handle instances one or a few at a time. Any system placed in a real-time learning environment will most likely have to deal with an incremental learning situation. Recent machine learning researchers have recognized the need for incremental learning systems. AQ15's incremental capability is a response to this perceived need. We now describe the incremental learning facility that occurs at STEP 4 of the Star algorithm presented in the previous section.

If AQ15's incremental learning facility is used, then after each subset of the instances is presented to the system, AQ15 forms a hypothesis that is consistent and complete with respect to all instances so far presented including the most recent subset. Each acceptance of a new subset of the instances is associated with a new iterative step of instance processing. To cut down the cost of induction when a new processing step begins, the previous hypothesis is passed to the Star algorithm. Therefore, rather than forming a new hypothesis from scratch by using all the old instances as well as the new instances, AQ15 forms a new hypothesis that covers the new instances and the previous hypothesis (from the previous iterative step). Of course, it is possible that one or more of the new hypothesis complexes is inconsistent with the instances. If this is the case, then the inconsistent complex must be *specialized* so that it no longer covers any instances that are classified as negative examples. Specialization is accomplished by adding selectors to a complex to make it more specific (i.e., less general).

Another important capability for a learning system to be applied to real-world applications is the ability to handle noisy data (i.e., ambiguous instances). Data is considered noisy whenever an instance is classified (by the teacher) as positive (respectively, negative), and then another instance with identical feature values for all features is classified as negative (respectively, positive). The source of the error could be attributable to either poor instruments for detecting the feature values or a teacher that generates faulty classifications. AQ15 can handle data that is somewhat noisy.

However, the level of noise tolerated before AQ15's performance degrades has not yet been published. Any learning system that requires its hypotheses to be consistent with all instances and that does not use statistical learning methods will probably have difficulty with large amounts of noise in the data. Of course, if the language is insufficient to describe the instances consistently, then the data will also appear noisy. The correct response to an insufficient instance language is to create new terms. Although AQ15 can create new terms by using constructive induction, it does not create new terms in response to inconsistently labeled instances.

AQ15 responds to inconsistent instance classifications (noise) with one of three approaches. The user can decide which approach the system should take. The three options are: (a) treat inconsistent examples as positive examples, (b) treat them as negative examples, or (c) remove them from the data set (Michalski et al. 1986).

In addition to noisy data, AQ15 can also handle incomplete data. There is one type of incomplete data that AQ15 can handle—namely, missing feature values in the descriptions of instances. The system cannot handle other types of incomplete data, such as a missing classification or a partially specified generalization tree.

Another way in which a system can adapt to realistic situations (besides being incremental and handling noisy data) is to allow a flexible representation of concepts. Many concepts have a basic, core definition along with a portion of the definition that may vary. For example, a chair always has a seat. However, the concept of a chair may be flexibly defined to include chairs with different numbers of legs as well as those with and without back supports. In response to the need for a flexible concept representation, POSEIDON, a successor of AQ15, has been developed (Bergadano et al.). POSEIDON uses a *two-tiered* concept representation consisting of both a static base concept and a second tier that allows flexibility in response to the current context. This second tier consists of matching procedures and inference rules that implicitly define the allowable context-dependent variations of the base concept.

Although AQ15 is not equipped to handle all realistic situations, there are a number of important domains in which it has already proven useful. The next two sections present the results of applying AQ15 to some of these domains. Successful results of applying POSEIDON to learn the concept of an acceptable union contract and the distinction between Democrats and Republicans in the United States Congress may be found in Bergadano et al.

To get an idea of the computational time required to process all instances of a natural database, we ran AQ15 on the breast cancer database, described next. This database has 286 instances. There are nine attributes, four of which are numeric, while five are nominal. The number of values for the nominal features ranges from two to five. The database is slightly noisy and has some missing feature values. There are two decision classes (concepts to be learned). AQ15 took approximately an hour to process all database instances on a Sun4 workstation.

SUMMARY OF EXPERIMENTAL RESULTS

Large databases often exist that contain cases of individuals with and without diseases and, therefore, concept learning from examples can be useful in applications involving disease diagnosis. Michalski, Reinke, and others have applied AQ15 to realistic domains where the system has learned

to characterize both human and plant diseases. The results are published in Michalski et al. 1986 and Reinke and Michalski 1988. The domains to be presented in this section are lymphography, breast cancer recurrence, locations of primary tumors, and soybean disease diagnosis.

The first set of experiments related to the lymphography, cancer, and primary tumor domains. In each case the system was run in batch mode and the instances were divided into a training set and a test set. Seventy percent of the database instances were considered training instances, and the remaining 30% were considered test instances. When AQ15 was applied to the training instances, it formulated a generalized hypothesis that was consistent and complete with respect to all the training instances. After learning a hypothesis on the training set, AQ15 was then applied to the test set to determine how well the system learned the concept. For each instance in the test set, AQ15 used its hypothesis (formed on the training set) to predict the classification (positive/negative) of that instance. AQ15 then found out from the database whether its perdiction was correct or incorrect. The percentage of correct predictions made by AQ15 on the test set was reported. Each experiment (training phase followed by testing phase) was repeated four times with a different (random) division of the examples between training and test sets each time. The results we give below of the percentage of correct predictions were averaged over all four experiments.* We now present AQ15's performance in each of these domains and compare it with the performance of (human) domain experts (Michalski et al. 1986).

The lymphography domain is characterized by 18 attributes and 4 decision classes (concepts to be learned). Data from 148 patients is in the database, which creates a total of 148 instances. AQ15's performance on the test set, after training, was 82% correct predictions. Although specialists have not actually been tested in this domain, a specialist in the domain estimated that internists diagnose correctly approximately 60% of the time and specialists diagnose correctly approximately 85% of the time.

Breast cancer reappears in about 30% of patients who undergo breast surgery. The ability to predict recurrence of breast cancer is important for both the patient and the doctor recommending further treatment. Data for 286 patients is available, and each instance associated with a patient is described in terms of nine attributes. Two decision classes can be learned: when cancer will recur and when it will not. The average of AQ15's correct predictions over four runs on the breast cancer database was 68%. The average of the prognoses of five specialists of the Institute of Oncology (University Medical Center, Ljubljana, Yugoslavia) was 64% correct. Therefore, in this domain, AQ15 was capable of outperforming a group of specialists.

In the primary tumor domain, 22 different decision classes (concepts) can be learned corresponding to the 22 possible locations of a primary tumor. Instances correspond to data from 339 patients, and 17 attributes are used to describe the characteristics. AQ15 demonstrated the capability to predict correctly 41% of the time in this domain. At the Institute of Oncology, four internists and four specialists were tested. The results of these tests were averaged. The correct location of the primary tumor was determined 32% of the time by the internists and 42% of the time by the oncologists.

The last domain we describe here is that of soybean disease diagnosis (Reinke et al. 1988). NEWGEM (Mozetic 1985) was used rather than AQ15 in these experiments. NEWGEM is the predecessor of AQ15. Differences between the two systems are of minor importance. Once again,

---

*Actually, AQ15 experiments were run with three different parameter settings, yielding slightly different results (Michalski et al. 1986). We only present the best results of the three settings.

the training instances were divided into a training and a test set. Learning on the training set was accomplished in incremental mode. Diseased soybean crops are described in terms of 50 attributes in this database. Attribute domains range from 2 to 11 values. Therefore, the space of possible attribute-value combinations is quite large. There are 17 different soybean diseases corresponding to 17 concepts to learn. Based on different LEF parameter settings, AQ15's performance varied from 75% correct to over 90% correct. Plant pathologists that were tested predicted correctly about 80% of the time.

Apparently, AQ15's performance can be competitive with that of experts on some realistic diagnosis domains. Further experiments, beyond those mentioned above, were performed at NCARAI to extend our understanding of the system. These experiments are described in the following section.*

NEW EXPERIMENTAL RESULTS

Previously in this report, we introduced the notion of bias for constraining the search in the machine learning system. Bias is highly important. The type of bias that is chosen to be implemented in a system can profoundly affect its performance. Therefore, the first set of empirical experiments that we performed was a comparison of four different concept learning systems in terms of the effect their bias had on their ability to learn concepts. We expected to learn how well NEWGEM performs in relation to other systems when its bias (e.g., parameter settings) is suitable for the data and target concepts to which the system is applied. NEWGEM was compared with one system (PREDICTOR, which was expected to be NEWGEM's toughest competitor in these experiments) that is also designed for maximum performance on these data and concept combinations, as well as with two other systems that are not as well suited for the given data.

Bias can be categorized into two major types: language bias and algorithmic bias. A language bias expands or restricts or in some way alters the hypothesis language, which effectively alters the hypothesis space. Constructive induction operators, which were described in the section on generalization operators above, are an example of a method for biasing the language. Algorithmic bias, such as the LEF used to constrain the search space in AQ15 and NEWGEM (as previously described here), prioritizes or creates subsets of hypotheses that have already been generated. In summary, language bias alters the hypothesis space whereas algorithmic bias affects the search within the space. Both methods can effectively reduce the search to make learning more tractable.

The Systems

The four systems that were used in these experiments are NEWGEM (Mozetic 1985), C4.5 (Quinlan unpublished), PREDICTOR (Gordon 1990), and an implementaiton of Iba's algorithm (Iba 1979) by Gordon called IACL (Gordon 1990). All four systems learn concepts from examples. We modified NEWGEM and C4.5 to run in an incremental batch (that we call *binc*) mode in order to obtain a fine-grained performance comparison between the systems.† When a system that is designed to do batch learning is run in binc mode, one instance at a time is accepted by the system. Each time a new instance is accepted, it is appended to the list of previous instances and the system is rerun in batch on all instances seen so far (i.e., all previous instances and the new instance). Here we briefly describe the systems, focusing primarily on their biases.

*The motivations for all experiments and their methodology described in the following section may be found in Gordon 1990.
†Incremental AQ15 and NEWGEM were not used because they were unavailable at the time of this study.

Because the (AQ) algorithm used in NEWGEM has been the topic of this report, the system need not be described at this point other than the LEF and other important parameter settings used during these experiments. Constructive induction was not used in these experiments. The algorithmic biases used by NEWGEM for these experiments require that the hypotheses be consistent and complete with respect to the instances. (All instances are unambiguous in the data sets that are used.) The LEF used was the default setting:

1. Maximize the number of newly covered examples (tolerance = 0.0).*

2. Minimize the number of features (tolerance = 0.0).

Recall that the second LEF requirement produces a system bias toward simplicity in terms of the length of the complexes (disjuncts). The maxstar parameter, which ranges from 1 to 100, was set to 20. Other parameters used had default values, including setting the trim parameter to favor simplicity (except during the "threat" database experiments, described below).

C4.5 is a concept learning system descended from the ID series (Quinlan 1986). The system generates hypotheses in the form of *decision trees* rather than logical expressions as used by NEWGEM and AQ15. These decision trees contain nodes and arcs. The nodes are associated with instance features and the arcs are associated with tests on the values of these features. Instances are classified with the decision trees by running the tests embedded within the decision tree and terminating with a classification as either positive or negative. Although the use of decision trees instead of logical expressions is a language bias, the effects of this particular bias are not studied in the experiments described here. See De Jong and Spears 1991 and O'Rorke 1982 for the results of experiments comparing these two hypothesis forms.

There are three algorithmic biases in C4.5. The first bias is an information theoretic measure that orders the test attributes (features) in the decision tree data structure used for hypotheses. This bias favors small decision trees. The advantage of small decision trees is that the computational expense of classification is reduced, and simplicity is gained. The second algorithmic bias of C4.5 is a tree pruning process. This bias also implements a preference for small decision trees. The third bias is the absence of a preference for hypotheses that are consistent with the instances. If strict consistency with the instances is not required, then the system is better able to handle noisy data. More details on ID-related systems may be found in the report Gordon and Spears (forthcoming).

Like AQ15 and NEWGEM and unlike C4.5, Iba's Algorithm for Concept Learning (IACL) (Iba 1979; Gordon 1990) forms DNF hypotheses. IACL learns incrementally by accepting one instance at a time. The system keeps a positive hypothesis to cover the positive examples and a negative hypothesis to cover the negative examples (similarly to AQ15 and NEWGEM). Each time a new instance is accepted, it may or may not be already covered by one of the hypotheses. If it is not covered, then the description of this instance is *merged* with each disjunct of the appropriate hypothesis (i.e., the hypothesis of the same class as the instance) to form new disjuncts for a new hypothesis.

---

*If the tolerance is 0.0, then this criterion must be met precisely.

During merging, a new disjunct is formed from each original disjunct and the description of the new instance. The new disjunct is more general than (i.e., implies) both elements from which it is formed. Generalization is minimized during merging. If the new instance is already covered by one of the hypotheses, then either it is correctly covered (i.e., it has the same class as the hypothesis), or it is incorrectly covered. If it is correctly covered, then no change is made to the hypotheses. If it is incorrectly covered, then the hypotheses are reformed to regain consistency. The algorithmic biases of IACL are preferences for consistent, complete, and specific (i.e., less general) hypotheses.

The version of PREDICTOR (Gordon 1990) that is used in these experiments has been constructed on top of IACL. Therefore, PREDICTOR also learns incrementally and maintains hypothesis consistency and completeness with respect to the instances. PREDICTOR differs from IACL, however, because it has an added ability to alter its hypothesis language (which can thereby also alter the hypothesis specificity preference). PREDICTOR is the only system out of the four described here that both focuses on the hypothesis language as its primary bias and also has the ability to dynamically alter its own hypothesis language. The other three systems use algorithmic biases. Furthermore, IACL and C4.5 have biases that cannot be changed, and NEWGEM's bias can only be changed prior to program execution.

The motivation for PREDICTOR's dynamic bias changes is to allow the system to appropriately adapt to whatever data is presented to the system. PREDICTOR actively requests instances to test its bias (i.e., its hypothesis language preference).* The part of the hypothesis language that is tested and altered by PREDICTOR is the set of features and feature values. The system can learn more quickly by selecting an optimal or nearly optimal subset of the original features and their values. For example, the system can weed out irrelevant features from the hypothesis language, thereby expediting the learning process.

IACL can use generalization trees to guide its generalizations. PREDICTOR can use generalization trees to guide its hypothesis language changes. For example, by using the trees of Fig. 1, PREDICTOR might decide that the distinction between aluminum and copper objects does not help the system to differentiate positive and negative instances. In that case, the system can remove this distinction from the language and thereafter describe all aluminum objects as having material "element," which is the parent value of "aluminum" and "copper." Generalization trees that are given to a system can either be "correct" or "incorrect." Correctness refers to the relationship between a tree and the target concept. When a system uses a correct tree, it is making correct generalizations. In other words, the generalizations it makes bring the hypotheses closer to the target concept. When a system uses an incorrect tree, it can be misled by the tree to make bad generalizations. Bad generalizations take the hypotheses farther away from the target concept. PREDICTOR has more to gain from correct trees and more to lose from incorrect trees than IACL because PREDICTOR changes its hypothesis language by using these trees, which is a more sweeping change than simply generalizing hypotheses as IACL does. PREDICTOR avoids the pitfalls of using bad generalization trees, however, by running tests prior to making hypothesis language changes and by making only those changes that pass the tests.

C4.5 is not designed to handle structured nominal features. NEWGEM can use generalization trees. Nevertheless, NEWGEM's use of the trees is cosmetic and is, therefore, does not change the system's learning performance.

---

*The implication of this ability to actively request instances is that PREDICTOR sees the instances in a different order than the other systems do. This ability may provide the system with an advantage over the other systems.

The Databases

Three databases were used in the experiments to be described. Two of the databases were artificially created, and the third is a variant of the natural soybean disease diagnosis database described previously in this report. None of the databases presented here contains any noise. In addition to describing the databases, this section also presents the target concept that was learned for each database. The target concepts presented here (that are used in the experiments) are all simple concepts. Simple concepts and nonnoisy data were chosen for these experiments because we considered these characteristics to be compatible with NEWGEM'S biases—namely, a preference for simple hypotheses (that was a user-chosen bias) and hypothesis consistency and completeness with respect to the instances.

The first artificial database (DB1) used consists of instances described by five features. The features are "size," which has three values; "material" and "shape," which both have five values; "texture," which has two values; and "color," which has 24 values. The concept that was learned (TC1) states that instances having any one of 12 specific color values are positive and instances of any other color are negative. All of the other features are irrelevant for distinguishing whether the instance will be classified as positive or negative.

All four systems used the same database. However, PREDICTOR used internal generalization tree data structures that allowed this system to interpret the features as structured nominals, whereas IACL, NEWGEM and C4.5 did not use these generalization trees. These trees are correct for learning target concept TC1. Figure 2 shows the trees for all features other than color. The tree for color has two children of the root node, and each child of the root has 12 children (i.e., the 24 color values).

The second artificial database that was used consists of instances described by using five numeric features. For this domain, the assumed task was to classify an incoming plane as threatening or nonthreatening. The features and their range of values and precision that were used are the following:

Feature "range:" range = [0,100], precision = 10
Feature "altitude:" range = [0, 60000], precision = 500
Feature "speed:" range = [500, 2000], precision = 100
Feature "bearing:" range = [0,360], precision = 10
Feature "heading:" range = [0,360], precision = 10

The concept that was learned, called "threat," was a simple conjunction of feature values. An airplane was considered threatening if its range was within the interval [20, 40] and its altitude was within the interval [500, 20000]. Otherwise, the airplane was not considered a threat.

The soybean disease database was simplified for the experiments described below. The original database contains 17 target concepts corresponding to 17 different soybean diseases. The single target concept chosen for this experiment classifies all instances with the cyst nematode (CN) disease as positive and all other instances as negative. The original database was edited to remove many irrelevant features, to remove noise, and to fill in missing feature values with arbitrary (randomly
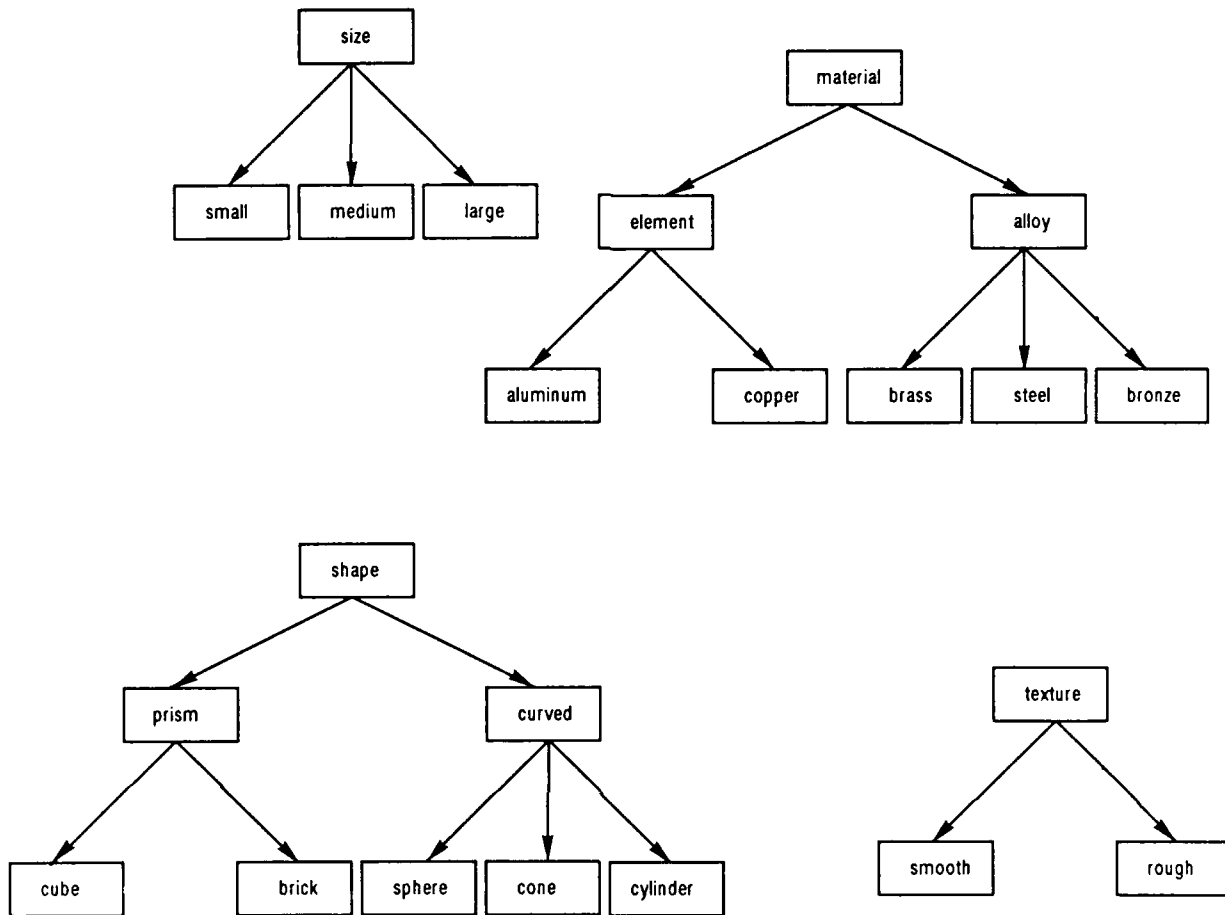
Fig. 2 — Generalization trees for database DB1

chosen) values. These edits were made because PREDIC ఎR cannot handle noise or missing feature values and, furthermore, many irrelevant features might complicate the performance comparisons. Each edit was made with the assumption that there could be a realistic justification for the edit. The final, edited database (as well as these justifications) is found in Gordon 1990.

The resulting database contains five features: "date," "plant stand," "precipitation," "crop history," and "area damaged." The features "date" and "precipitation" have seven values; "plant stand" has two values; "crop history" has four values; and "area damaged" has six values. PRE-DICTOR and IACL used generalization trees (structure) with all features other than "crop history." C4.5 and NEWGEM did not use generalization trees. The feature "crop history" has values {diff-last-year, same-last-year, same-last-two-years, same-last-several-years}. Figure 3 shows the trees for the other features. The structure that was added to features in the soybean database was partially misleading (i.e., some of the added generalization trees are not "correct" for the CN target concept). The concept that was learned is the following:

(date = summer) and (crop history = same-last-two-years) and (area damaged = lower) OR
(date = summer) and (crop history = same-last-several-years) and (area damaged = upper).
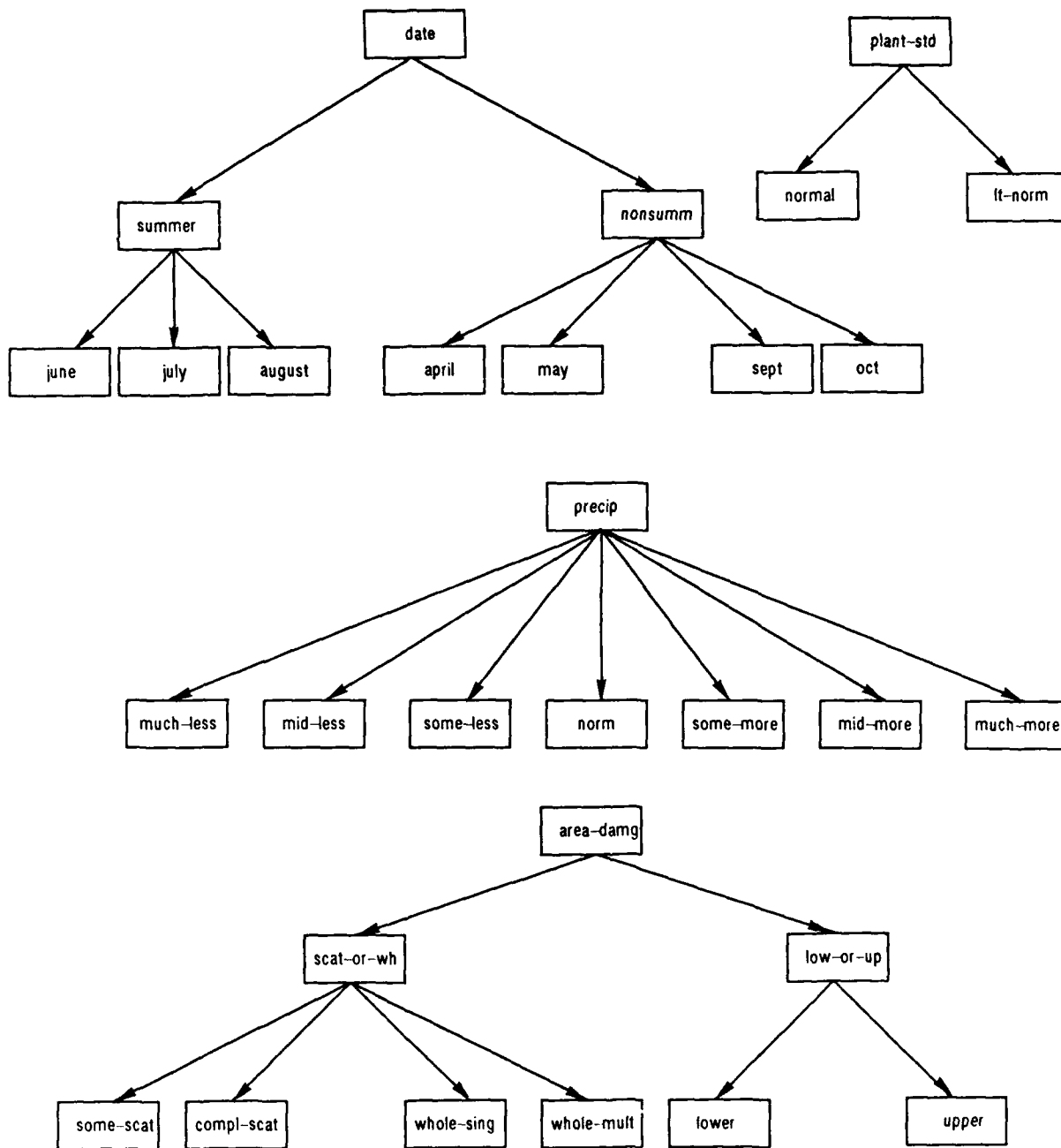
Fig. 3 — Generalization trees for learning the soybean target concept

Empirical Results

Here we describe results of running each of the four systems on each of the three databases discussed above. NEWGEM and C4.5 were run in binc mode, whereas PREDICTOR and IACL used incremental learning. When using the two artificial databases, PREDICTOR requested the instances it received. The other three systems were run on randomly generated sets of instances. When using the soybean database, all four systems used the same randomly generated set of instances. PREDICTOR had access to ''extra'' requested instances on the soybean database for resolving errors in its bias selection process. See Gordon 1990 for details.

The systems were run on a cycle consisting of receiving a new instance, predicting the class of the instance, getting feedback about the correctness of the prediction, then updating the current hypothesis appropriately to incorporate the new instance. Predictions were made by using the system's current hypotheses. Figure 4, the performance graph, depicts the percentage of correct predictions made by the systems over time (where time is incremented with each new instance). Every data point on the graph represents average performance over 10 runs. Furthermore, each point represents the percentage of correct predictions over a window of previous instances. The window size is 10.
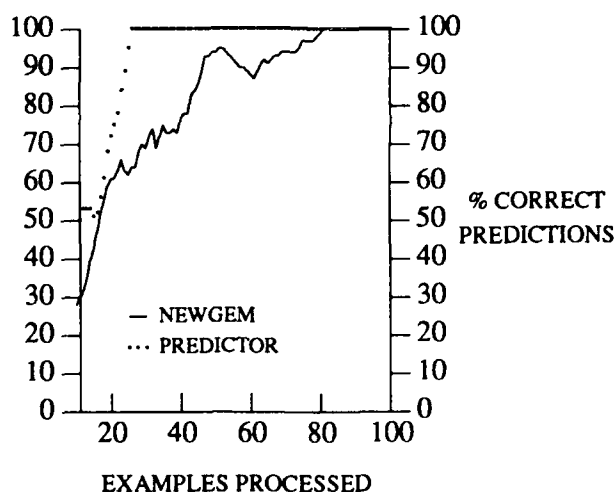

Fig. 4 — TC1: NEWGEM and
PREDICTOR

The performance of system pairs can be compared on the basis of two measures. The first measure is how long it takes the systems to converge on the final, correct concept. In other words, this is how many instances the systems require to learn the final target concept. The second measure is prediction accuracy (i.e., the shapes of the curves). The reason that two measures were used in these experiments is that one or the other may be more important, depending on the application. For example, the convergence rate may be more important if a concept is learned easily. On the other hand, in some situations prediction accuracy might be more important. For example, some target concepts are very complex and unlikely to ever be learned given the sparse sampling of instances in the database. If this is the case, then the percentage of correct predictions on the instances seen so far is the important measure.

GORDON AND SPEARS

The first set of results compares NEWGEM with the other three systems on DB1 learning TC1. Figures 4 through 6 show that PREDICTOR performed better than the other systems when learning TC1 on DB1 both from the standpoint of convergence to the target concept and prediction accuracy. Although IACL had better prediction accuracy than NEWGEM for this target concept, NEWGEM had a better rate of convergence to the target concept; this was verified in the execution traces because 100% accuracy does not necessarily imply convergence to the final concept. NEWGEM was superior to C4.5 in terms of both prediction accuracy and convergence rate on this database and target concept (after the first 20 instances were seen).
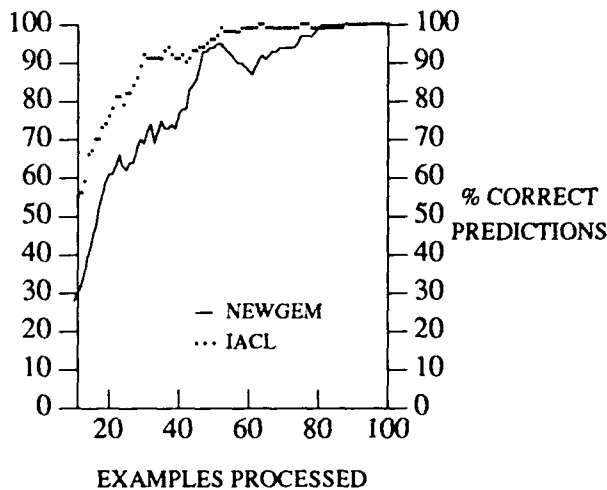
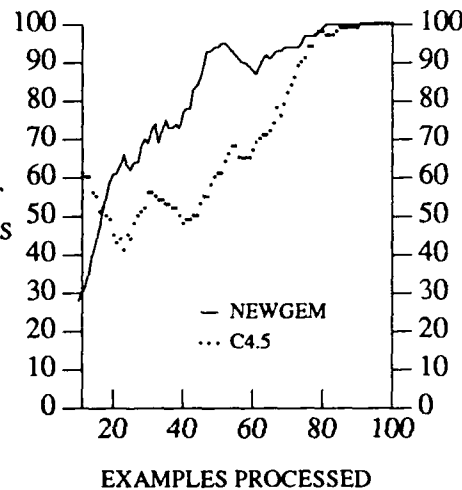Fig. 5 — TC1: NEWGEM and IACL

Fig. 6 — TC1: NEWGEM and C4.5

PREDICTOR's advantage from using generalization trees as well as its bias testing and adjustment capability were responsible for its outstanding performance learning TC1 on DB1. This is a simple database and target concept and is therefore well-suited for PREDICTOR, which can select a subset of the original hypothesis language to use for converging rapidly on the target concept. Although NEWGEM has neither of the advantages that PREDICTOR has, the system performed quite competitively. It far outperformed C4.5 after seeing only 20 instances. NEWGEM's algorithmic biases favoring shorter complexes (disjuncts) and hypothesis consistency and completeness with the instances worked well with this simple target concept and nonnoisy data.

The reason that C4.5 did not perform as well as NEWGEM is that C4.5 pruned its decision trees prematurely. C4.5 pruned its decision trees from the beginning and, because consistency was not enforced, the system developed a tree that was overly simplistic, rejecting what it considered to be noise. Eventually, however, it stopped pruning and recovered to learn the final concept. NEWGEM was more conservative in its learning than C4.5. Although NEWGEM preferred simpler hypotheses, its bias toward consistency with the instances prevented the system from mistakenly assuming that aberrant instances were noise.

20

The second set of results compares NEWGEM's performance with that of the other three systems on the threat database target concept. Figures 7, 8, and 9 show the graphs contrasting NEWGEM's performance with that of the other three systems. (Note that vertical axes begin at 50% in these and all remaining figures.) The systems were run on 300 instances. NEWGEM was used with a slightly different parameter setting. Rather than using a setting for the trim parameter that expresses a preference for simpler hypotheses, a setting that produces a preference for more general hypotheses was used. This is the only experiment for which the parameter settings differed in NEWGEM. The reason for the change in this experiment was that a suggestion was made (Zhang, personal communication) that this is an appropriate setting for the trim parameter in NEWGEM for learning this particular concept. Initial experiments confirmed the appropriateness of this setting. Note that initial experiments and forethought are sometimes required by the user to select a good bias for NEWGEM.
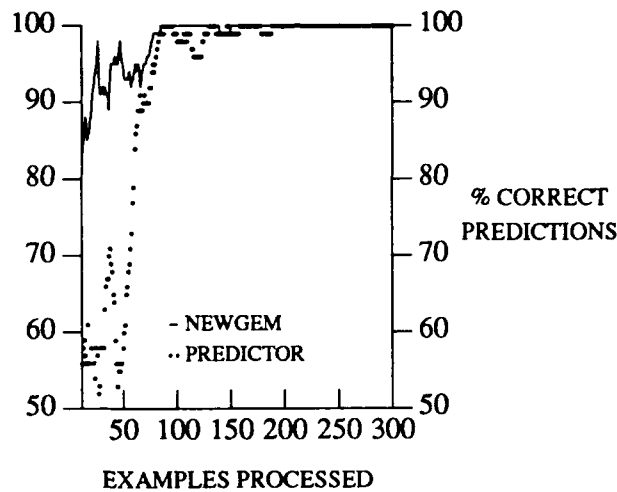
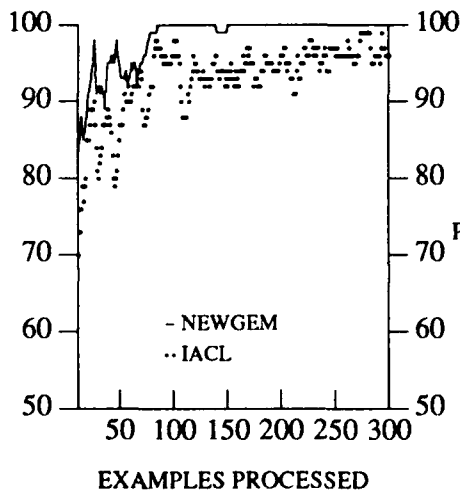

Fig. 7 — Threat: NEWGEM and
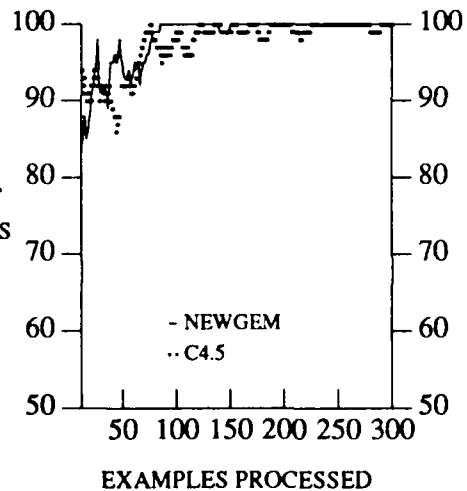PREDICTOR

Fig. 8 — Threat: NEWGEM and
IACL

Fig. 9 — Threat: NEWGEM and
C4.5

21

After consulting the execution traces, it was determined that none of the systems except PREDICTOR learned the threat target concept. This is because all systems other than PREDICTOR were subject to random input. PREDICTOR, on the other hand, actively requested instances. Nevertheless, PREDICTOR paid a high penalty (in terms of its prediction accuracy) during the first 100 instances for its active learning to find this concept. In terms of prediction accuracy, NEWGEM's performance was superior to that of all the other systems. Figure 7 shows the curve of NEWGEM achieving 100% correct approximately 50 instances earlier than PREDICTOR achieved this percentage. Therefore, based on the curves, NEWGEM performed better than PREDICTOR. IACL performed the worst in all respects when learning this target concept.

For the "threat" concept, for which a database with numeric features was used, PREDICTOR did not have an advantage of using generalization trees. Instead, its superior convergence rate was obtained by the system's methodical active learning. PREDICTOR performed a binary search of the feature space. Despite the fact that NEWGEM did not select its instances, the system was able to achieve a high degree of prediction accuracy. This was due to the appropriateness of the parameter settings, which favored generality, consistency, and completeness. A bias toward simplicity was helpful to C4.5. For this target concept, the lack of a consistency requirement did not significantly hinder C4.5's prediction accuracy. The biases of IACL were least suitable for this concept. Its bias toward specific hypotheses caused more problems on numeric data than the biases of the other systems.

The final set of results is a comparison on the soybean database learning the target concept CN. Figures 10 through 12 show the performance of NEWGEM in comparison with the other three systems.
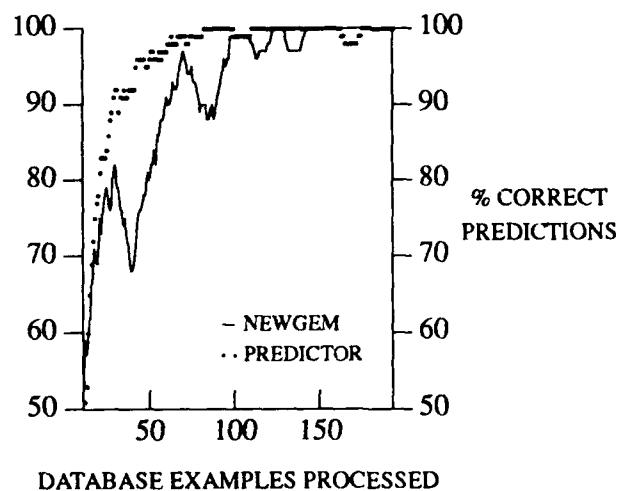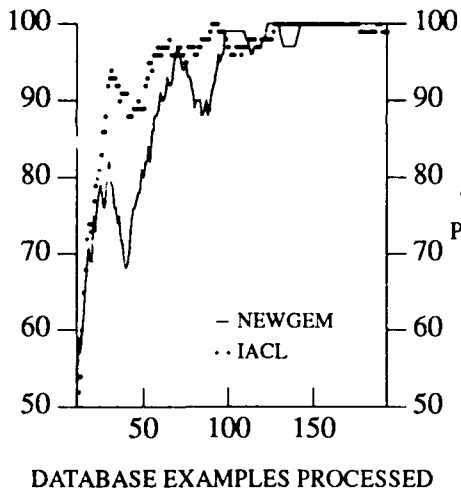

Fig. 10 — CN: NEWGEM and
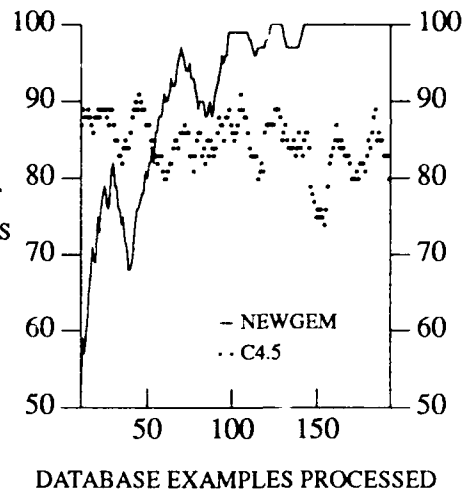PREDICTOR

Fig. 11 — CN: NEWGEM and
IACL

Fig. 12 — CN: NEWGEM and
C4.5

On the soybean database learning the CN concept (i.e., a description of soybean plants having this disease but not any of the other diseases), NEWGEM converged on the target concept sooner than any of the other systems. NEWGEM learned the concept after an average of approximately 140 database instances whereas PREDICTOR learned the concept after an average of approximately 175 database instances. Neither IACL nor C4.5 ever converged on the correct concept, although IACL performed well in terms of prediction accuracy. Therefore, in terms of prediction accuracy, PREDICTOR performed the best, then NEWGEM, IACL, and finally, C4.5.*

Both NEWGEM and PREDICTOR performed well because the CN target concept is simple, and the soybean data is not noisy. IACL, as was typical in all of these experiments, performed reasonably well but not outstandingly. This is because IACL tends to be a good overall performer but is not as well suited to simple target concepts as the other systems. Its bias is toward more specific, rather than simpler, hypotheses. Once again, C4.5's pruning mechanism prevented the system from learning the concept from the given instances. Whereas simplicity was increased by pruning, accuracy was decreased. After a while, the pruned tree predicted negative on every instance. The positive instances were considered noisy. This policy allowed the system to achieve between 80 and 90% correct; however, it prevented the target concept from being learned.

A two-tailed, t-test checking at a 95% significance level (i.e., $P \geq 0.05$) confirmed the significance of the comparison of the prediction accuracy of the system pairs shown in all graphs. The only pair of curves for which statistical significance did not hold over a substantial interval (i.e., 15 or more examples) is the pair of curves shown in Fig. 9 when comparing NEWGEM and C4.5 on the threat database. Most of the pairs of curves were significant for intervals of 30 examples or more.

In summary, we have selected a set of databases and target concepts that is expected to be well suited to the system NEWGEM and its (partially user-selected) biases. NEWGEM performed well under these circumstances. The experiments on these databases demonstrated NEWGEM's excellent performance in comparison with other systems. NEWGEM performed competitively with PREDICTOR, a system that is also designed for learning simple, nonnoisy concepts. Furthermore,

---

*PREDICTOR's success is partially due to its use of extra instances (not graphed here) that are not in the database.

NEWGEM often outperformed both IACL and C4.5 on these problems. The reader should keep in mind that the system parameter settings that we selected were suitable for the target concepts used here. Different settings would make NEWGEM suitable for different types of data and target concepts. In the future, we would like to run more experiments to determine how much system performance suffers when there is a mismatch between NEWGEM's parameter settings and the data and target concept. Furthermore, we would like to run experiments with different parameter settings for NEWGEM than the ones described here. We hope that these experiments will reveal more of the strengths and limitations of the NEWGEM system so that potential users will be better informed. If the user knows the characteristics of the database and can roughly estimate (based on initial experiments) target concept characteristics, then the information provided in this and future reports about system performance should be helpful.

SUMMARY

We have presented an overview of the capabilities of the machine learning system AQ15 and related systems. We have also presented some results on realistic domains. The intention of this and subsequent reports on other machine learning systems is to familiarize people working in various Navy applications with the characteristics, strengths, and limitations of these systems. The potential for a match between systems and applications can then be assessed. The more machine learning systems are applied to realistic domains, the more we can expand our knowledge of the type of domains for which they are most useful.

ACKNOWLEDGMENTS

We thank Jianping Zhang and Ryszard Michalski for providing copies of the NEWGEM and AQ15 programs. We also thank Ross Quinlan for providing a copy of the C4.5 program.

REFERENCES

F. Bergadano, S. Matwin, R. Michalski, and J. Zhang, "Learning Two-Tiered Descriptions of Flexible Concepts: The POSEIDON System," *Machine Learning* (forthcoming).

K. De Jong and W. Spears, "Learning Concept Classification Rules using Genetic Algorithms," in *Proceedings of the International Joint Conference on Artificial Intelligence*, Australia, Aug. 1991.

D. Gordon and W. Spears, "Machine Learning Systems: Part II—Concept Learning from Examples with ID3 and Related Systems," NRL Report forthcoming.

D. Gordon, "Active Bias Adjustment for Supervised Concept Learning," Ph.D. dissertation, Department of Computer Science, University of Maryland, College Park, MD, 1990.

A. Hamilton, *Logic for Mathematicians* (Cambridge University Press, London, 1978).

J. Hong, I. Mozetic, and R. Michalski, "AQ15: Incremental Learning of Attribute-based Descriptions from Examples, the Method and User's Guide," Univ. of Ill. Tech. Rep. Num. ISG-86-5, 1986.

G. Iba, "Learning Disjunctive Concepts from Examples," Mass. Inst. of Tech. A.I., Memo 548, 1979.

R. Michalski, "A Theory and Methodology of Inductive Learning," in *Machine Learning: An Artificial Intelligence Approach*, Vol. 1, R. Michalski, J. Carbonell, and T. Mitchell, eds., (Tioga Publishing Co., Palo Alto, CA, 1983), Ch. 4.

R. Michalski, I. Mozetic, J. Hong, and N. Lavrac, "The AQ15 Inductive Learning System: An Overview and Experiments," Univ. of Ill. Tech. Rep. Num. ISG-86-20, 1986.

T. Mitchell, "The Need for Biases in Learning Generalizations," Rutgers Univ. Tech. Rep. Num. TR CBM-TR-117, 1980.

I. Mozetic, "NEWGEM: Program for Learning from Examples, Program Documentation, and User's Guide," Univ. of Ill. Rep. Num. UIUCDCS-F-85-949, 1985.

N. Nilsson, *Principles of Artificial Intelligence* (Tioga Publishing Co., Palo Alto, CA, 1980).

P. O'Rorke, "A Comparative Study of Inductive Learning Systems AQ11P and ID-3 Using a Chess Endgame Test Problem," Univer. of Ill. Rep. Num. UIUCDCS-F082-899, 1982.

F. Pipitone, K. De Jong, W. Spears, and M. Marrone, "The FIS Electronics Troubleshooting Project," in *Expert Systems Applications to Telecommunications*, J. Liebowitz ed., (Wiley and Sons, Inc., New York, 1988), pp 73-101.

J. Quinlan, "Induction of Decision Trees," *Machine Learning* 1 (1), 81-107 (1986).

J. Quinlan, "Documentation and User's Guide for C4.5," (unpublished).

R. Reinke and R. Michalski, "Incremental Learning of Concept Descriptions: A Method and Experimental Results," in *Machine Intelligence*, Vol. 11, J. Hayes, D. Michie, and J. Richards eds., (Oxford Press, New York, 1988).

J. Zhang, George Mason University, personal communication, 1990.