Machine Learning, 20, 35–61 (1995)

Inductive Policy: The Pragmatics of Bias Selection

FOSTER JOHN PROVOST foster@nynexst.com
NYNEX Science & Technology, White Plains, NY 10604

BRUCE G. BUCHANAN
Intelligent Systems Laboratory, Computer Science Department, University of Pittsburgh, Pittsburgh, PA 15260

Editor: M. desJardins and D. Gordon

Abstract. This paper extends the currently accepted model of inductive bias by identifying six categories of bias and separates inductive bias from the policy for its selection (the *inductive policy*). We analyze existing "bias selection" systems, examining the similarities and differences in their inductive policies, and identify three techniques useful for building inductive policies. We then present a framework for representing and automatically selecting a wide variety of biases and describe experiments with an instantiation of the framework addressing various pragmatic tradeoffs of time, space, accuracy, and the cost of errors. The experiments show that a common framework can be used to implement policies for a variety of different types of bias selection, such as parameter selection, term selection, and example selection, using similar techniques. The experiments also show that different tradeoffs can be made by the implementation of different policies; for example, from the same data different rule sets can be learned based on different tradeoffs of accuracy versus the cost of erroneous predictions.

Keywords: inductive learning, inductive bias, bias selection, inductive policy, pragmatics

1. Introduction

Inductive learning takes place within a conceptual framework and set of assumptions (Buchanan, Johnson, Mitchell, & Smith, 1978). The term *inductive bias* refers to the choices made in designing, implementing and setting up an inductive learning system that lead the system to learn one generalization instead of another (Mitchell, 1980). Unfortunately, different domains, tasks, and situations may require different inductive biases. We present a model of inductive learning that includes not only an explicit bias, but the additional considerations involved in representing and selecting an inductive bias. These extra considerations, which deal with relationships between the learning program and the user, such as the costs of learning and using incorrect relationships, we term *pragmatic considerations.*

An *inductive policy* is a strategy for selecting inductive bias based on a task's pragmatic considerations.[1] An inductive policy addresses tradeoffs to be made in a particular domain. For example, higher accuracy achieved by looking at more examples may be traded for the ability to learn within a given resource bound. There are many dimensions to inductive bias, so an inductive policy may be considered to select biases by searching a multi-dimensional space. A satisfactory bias is one that satisfies some criteria defined by the pragmatic considerations, both in terms of the performance of the learned concept description (e.g., with respect to accuracy and error cost) and in terms of the performance of the learning system (e.g., with respect to resource constraints).

We show that not only is it important to represent *inductive bias* explicitly, but that a framework where *inductive policy* is represented explicitly can effectively represent different policies for selecting bias under different circumstances. Types of bias selection previously considered separately can be treated uniformly within this framework, e.g., parameter selection, term selection, and example selection. A system built around this framework will be flexible; it will be able to obtain different behavior from the learning

system—even on the same training data—based on different considerations. In particular, we address pragmatic considerations of space, time, accuracy, and error cost.

2. Inductive Bias & Inductive Policy

The dual-space model of induction proposed by Simon and Lea (1974) makes explicit the rule and example spaces (s_r and s_e in Fig. 1), the methods of searching the spaces (m_r and m_e), the relations between the examples and m_r ($\rho_{e,r}$), and the relations between the rules and m_e ($\rho_{r,e}$). The relations, $\rho_{e,r}$ and $\rho_{r,e}$, correspond to the ways in which the rules and examples interact in learning; e.g., examples are used to evaluate rules, and rules can be used to guide intelligent example selection. In principle, any of these can be changed independently of the others. For example, the same method (m_r) can be used to search the rule space with different ways of using the examples to evaluate the rules ($\rho_{e,r}$).[2]

2.1. *An Extended Model of Inductive Bias*

We add to this model an explicit inductive bias, which includes choices that instantiate all of the specifiable parts of the Simon and Lea model: s_r, s_e, m_r, m_e, $\rho_{e,r}$, and $\rho_{r,e}$. For any particular learning system, some of these choices may be built in. However, many learning systems allow one to change the bias along one or more of these six dimensions. For example, most learning programs allow one to change the set of terms used to describe hypotheses.

The space of different inductive biases has been partitioned into choices that restrict the space of hypotheses considered (*restriction* biases) and choices that place an ordering on the space of hypotheses (*preference* biases) (Dietterich, 1991; Rendell, 1986; Utgoff, 1986). Very little work discusses bias relating to the example space (as such); one exception is the work of Ruff (1990). In Section 5.3, below, we describe how a system can deal with pragmatic considerations involving the example space using the same techniques used for bias selection along other dimensions.

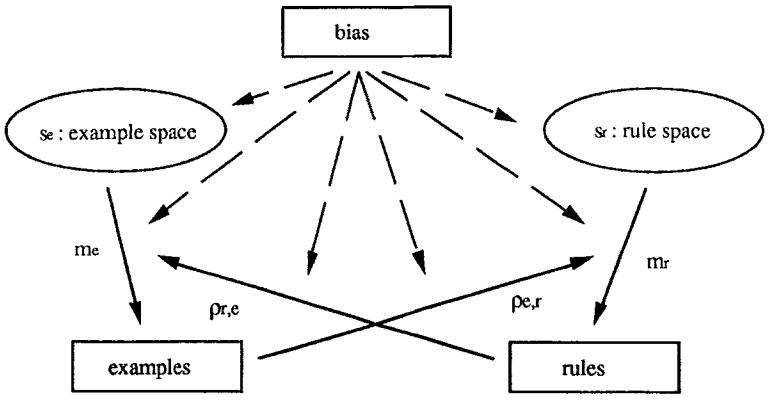

Figure 1. Inductive bias defines the rule and example spaces (s_r and s_e), the methods of searching the spaces (m_r and m_e), the relations between the examples and m_r ($\rho_{e,r}$), and the relations between the rules and m_e ($\rho_{r,e}$). Solid arrows indicate relationships emphasized in the original model of induction as a problem of searching two spaces (Simon & Lea, 1974).

Biases may also be partitioned according to the part of the learning problem that they affect. Rendell, et al. (1987) partition biases into *representational* manifestations and *algorithmic* manifestations. Representational manifestations include constraints on the description of objects (e.g., hypotheses). These correspond to the definitions of the rule space and can be extended to the example space (s_r and s_e). Algorithmic manifestations include constraints on the construction, transformation and verification of hypotheses. These would then correspond to the methods for searching the spaces (m_r and m_e). Most learning programs have parameters with which one can change the representational or algorithmic bias.

2.2. *Inductive Policy*

Although the concept of inductive bias has been analyzed and formalized (Haussler, 1988; Rendell, 1986; Utgoff, 1984), there has not been a clear distinction between the bias choice and the *strategy* for making a bias choice. To avoid confusion, we call a choice of specific s_r, m_r, s_e, m_e, $\rho_{e,r}$, or $\rho_{r,e}$ a *bias choice*. *Inductive policy* denotes the strategy used for making bias choices. Inductive policy decisions involve addressing tradeoffs and thus guide the selection of a suitable bias. These decisions are based on underlying assumptions of each domain, of the learning goals within the domain, and of each particular investigator. Bias comprises syntactic and semantic choices; the inductive policy addresses *pragmatics*—the *relationship between the bias choices and the goals of the people using the learning program.* In Section 4 we discuss how inductive policies can be represented explicitly in a system.

An example of underlying assumptions in a domain concerns the total number of examples available for learning and the amount of processing power. With many examples and little computing power, the machine learning researcher immediately wants to consider different strategies from those he or she would consider if the opposites were true, e.g., (i) use a relatively inexpensive algorithm, (ii) use a technique for selecting a subset of examples (intelligent or random), or (iii) use an incremental learning method.

Just as early learning systems were constructed with an implicit fixed bias, more recent learning systems have an implicit fixed policy for selecting bias when they include bias selection at all. In Section 3 we analyze existing systems with respect to the policies they use for searching their bias spaces. We identify three techniques that are used in many inductive policies, which we will call *inductive policy techniques*. These techniques are the beginnings of a tool kit for building inductive policies for new problems. We simplify by describing a bias as either sufficient or insufficient, where "sufficient" means that a satisfactory concept description can be learned with that bias.

2.3. *Techniques for Building Inductive Policies*

By analyzing existing systems for selecting inductive bias, we can identify techniques that have been used in their inductive policies. Three techniques are commonly used and are not specific to any one category of bias:

- adding structure to the bias space
- using learned knowledge for bias-space search guidance
- constructing a learned theory across multiple biases

Adding structure to the bias space involves adding detail to any or all of the six categories of bias (Fig. 1), using domain knowledge and knowledge regarding various biases to order or to restrict the bias-space search.[3] For example, consider a problem where there are too many features for feasible learning within a limited amount of space. Suppose we are relatively sure that there will be many (simple) redundant, satisfactory descriptions. We can then restrict the search to consider only feature subsets of size k, where k is small, and still have a good chance of finding a sufficient bias. Other work has considered optimal policies for ordering search biases based on cost and knowledge regarding probable sufficiency (Slagle, 1964; Simon & Kadane, 1975; Korf, 1985; Provost, 1993).

The second technique for building inductive policies is to use the results of learning with the first i biases to determine an appropriate bias $i + 1$. This assumes that new biases can be formed from existing biases either by altering them or by combining elements of several previous biases, and that an evaluation function can be defined such that biases that give high scores often lead to biases that are sufficient. Let us return to our example of term selection, but without assuming much redundancy among the large number of terms. Let us assume that a restriction has been placed on the bias space (for space reasons, as described above) that limits bias selection to subsets of size k (for small k) from a very large set of possible terms. This bias space may be very sparse if there is no information about possible relevance of the various terms. One simple method of guiding the search of the bias space is to keep track of which terms were used in partial concept descriptions that perform well (e.g., the best so far), and add these to subsequent biases, hopefully increasing the probability that a subsequent bias will be sufficient.

A third technique is to build a concept description from elements learned in several biases, which we call *theory construction across biases*. Using such a technique assumes that (i) coherent concept descriptions can be constructed from partial concept descriptions; (ii) these partial concept descriptions can be evaluated for probable goodness, and (iii) partial concept descriptions that evaluate well often lead to concept descriptions that are satisfactory (as in rule learners (Clearwater & Provost, 1990; Michalski, Mozetic, Hong, & Lavrac, 1986)). An important side effect of this type of technique is that the partial concept description(s) constructed through bias i can be used as information to guide the bias-space search. In addition, learning with bias $i + 1$ can be restricted to address incompleteness and inconsistencies with respect to the currently held concept description; such *inductive strengthening* techniques are described elsewhere (Provost & Buchanan, 1992b).

This set of inductive policy techniques is not meant to be complete, but instead to represent techniques commonly used for bias selection. One technique that is not as common is to use the data directly to help guide bias selection. For example, term selection systems can use statistical tests to estimate the degree of relevance of each term. We will now review existing systems in light of the inductive policy techniques described above; the review provides a number of examples of each of the three techniques.

3. Implicit Inductive Policies

Addressing inductive policy is not new. Probably the first system recognized as an AI machine learning system was Samuel's checkers program (developed 1947–1967) (Samuel, 1963). Samuel studied several methods of learning; one method involved tuning a polynomial evaluation function for use in an alpha-beta minimax search. The checkers program

performed bias selection in the form of *term selection.* Samuel decided on a representation that would prohibit him from expressing interactions among the terms, but would provide him with a simpler representation. Later he made a different tradeoff, choosing the more complicated signature tables that would allow him to represent interactions among terms.

Samuel's program searched the bias space with respect to sets of terms (s_r). The bias selection policy began as random selection. The program constructed the concept description as it searched the bias space, as opposed to starting from scratch with the new set of terms, and used this description to help guide the selection process—the term making the least contribution was replaced.

Another early learning program, a pattern-recognition program developed by Uhr and Vossler (1963), used operators to transform an input pattern into a list of numerical characteristics, then compared these characteristics to those in memory associated with stored patterns. The features used to describe the patterns were the operators and their characteristics, both of which were created dynamically.

The program selected its bias dynamically with respect to s_r. It searched the bias space using both random selection and data-driven guidance based on input patterns received. In addition, the search was guided by the knowledge learned during the bias-space search: terms that performed poorly were discarded; terms that performed well were used to create new terms. The program constructed its concept description as it searched the bias space.

3.1. Bias Selection Systems

The first implemented[4] work to address explicitly the question of automatic bias selection was Utgoff's (1984) STABB (Shift To A Better Bias) system. It adds new terms to the concept description language based on two heuristics—*least disjunction and constraint back-propagation.* STABB operates on top of the LEX system (Mitchell, Utgoff, & Banerji, 1983), which uses the *candidate elimination algorithm* (Mitchell, 1978). The least-disjunction procedure is a goal-free method; terms are added by finding the least-specific disjunction of existing useful terms. The constraint back-propagation procedure introduces terms by deducing the domain of an operator sequence that proved to be useful (cf. Explanation-Based Generalization (Mitchell, Keller, & Kedar-Cabelli, 1986), Explanation-Based Learning (DeJong & Mooney, 1986)). After a shift of bias, STABB reprocesses all the training examples (i.e., the version spaces for all existing concepts are recomputed). STABB is one of several systems that construct new terms and add them to the description language (*constructive induction* (Dietterich & Michalski, 1983)—Matheus (1989) provides a comprehensive overview). Constructive induction differs from term selection, as in Samuel's program, in that the set of possible terms is not predefined extensionally.

In general, constructive induction systems search a bias space of sets of terms (s_r). Generally, these systems guide their bias-space search based on the learned rules, often based on knowledge of what seems to be working or not working. Some constructive induction systems perform theory construction across biases, others do not. STABB (and FRINGE (Pagallo, 1989) and CITRE (Matheus, 1989)) starts each search of the hypothesis space from scratch, not constructing a concept description across biases. STAGGER (Schlimmer, 1987), on the other hand, constructs its concept description as the system's bias is modified by adding new terms.

The VBMS system (Rendell, Seshu, & Tcheng, 1987) chooses a learning algorithm based on problem characteristics, such as the number of features and the number of examples. VBMS begins with no knowledge of the appropriateness of biases. Gradually, it induces relationships between problem classes and biases. VBMS clusters algorithms based on their past performance; when it is faced with a new problem, the system selects the "best" algorithm and tries it. If this algorithm does not perform satisfactorily, VBMS tries the next best and so on. When VBMS runs out of "good" candidates it picks algorithms randomly and tries them until all algorithms are exhausted. After this procedure is complete, VBMS updates its problem/algorithm performance statistics.

VBMS considers bias selection with respect to three learning programs, combining selection of m_r, $\rho_{e,r}$, and $\rho_{r,e}$. VBMS' policy initially is a randomly ordered exhaustive search of the coarse-grained bias space. As the bias space is searched (for different problems) VBMS gradually changes the orderings of the learning programs, and thus the structure of the bias space. VBMS performs no bias-space search guidance based on the learned results of the current problem, and does not perform theory construction across biases. The main contribution of this work is that VBMS performs "meta-level" learning, inducing what biases are appropriate for what problems, given that appropriate dimensions of the problem space can be identified.

Tcheng, Lambert, Lu and Rendell (1989) designed the Competitive Relation Learner (CRL), a generalized recursive splitting algorithm (cf. CART (Breiman, Friedman, Olshen, & Stone, 1984) and ID3 (Quinlan, 1986a)) that includes multiple decomposition strategies, multiple function-approximation strategies, and multiple decomposition-evaluation functions. CRL produces a hybrid representation of the learned concept by first estimating the quality of the learning strategies on the entire example set, and then seeing if decomposition can help matters by estimating the quality of the learning strategies after each type of decomposition. CRL is combined with a bias-optimization program called the Induce and Select Optimizer (ISO). ISO first probes randomly in the bias space, evaluating each probe by forming CRL's hypothesis with the given bias. ISO then attempts to describe an objective surface over the bias space and uses the examples and this surface to guide selection of the next bias with which to learn. This process continues until some stopping criteria are met.

CRL/ISO considers bias selection with respect to function approximation strategies (s_r), decomposition strategies, and other instantiations of $\rho_{e,r}$ and $\rho_{r,e}$. ISO's policy for bias selection begins as random selection of biases. The results of learning with the various biases are used for bias-space search guidance. CRL/ISO can produce a set of Pareto optimal[5] biases with respect to different objectives, thereby addressing different pragmatic considerations as a pre-process to learning. However, it is limited in the categories of bias it addresses and its strategy for searching the bias space is fixed; policies taking resource constraints on *learning* into account, for example, could not be represented.

In more recent work, Brodley's (1993; this volume) Model Class Selection (MCS) system also forms a tree-structured hybrid classifier that chooses between or mixes three representations: linear discriminant functions, standard decision trees, and instance-based classifiers. MCS is shown to be robust. Over a large collection of data sets, the system never performs much worse than the best of the primitive learning components, and in some cases out-performs them all.

MCS searches a bias-space of model classes (s_r), and data-partitioning functions ($\rho_{e,r}$ and $\rho_{r,e}$). The search of the bias space is guided by heuristic rules that examine characteristics

such as the number of features describing the data, the number of instances, the performance of the learned classifier, its compression, and the information gain. The hybrid classifier is constructed as the bias space is searched, and the space can be structured with a global bias, which prefers one type of function to the others in certain situations.

Gordon (1990) addresses active bias selection. Specifically, she uses actively requested examples to test explicit "biasing assumptions," and subsequently to select appropriate "bias adjustments." These biasing assumptions are defined as beliefs that a particular attribute, attribute value, or attribute value combination is unnecessary for learning the target concept. When a biasing assumption is tested and found to be suitable or not, the selection of terms follows in order to incorporate or delete that constraint.

Gordon's system, PREDICTOR, adjusts its bias with respect to s_r. PREDICTOR constructs its concept description as it adjusts its inductive bias. It guides its search of the bias space by the currently held concept descriptions and the actively requested training examples.

More recently, Spears and Gordon (1991) have explored the use of genetic algorithms (GAs) to produce a multistrategy concept learner that can select adaptively from among alternative learning methods. Their system, adaptive GABIL, searches the space of methods and the space of hypotheses in parallel. They consider adding two alternative methods to their GA-based concept learner: (i) dropping a feature from a disjunct, and (ii) adding a (internal) disjunct to the current classification rule.

Adaptive GABIL's bias space consists of two binary dimensions (m_r). It constructs its concept description as the system searches the bias space, and the search is guided by the partial concept descriptions formed along the way.

Holder (1990) also addresses multistrategy learning. As with VBMS, his PEAK system addresses three learning methods: rote learning, ID3, and EBL, combining m_r, $\rho_{e,r}$, and $\rho_{r,e}$. However, PEAK constructs its concept description across biases. The bias space is structured based on prior knowledge (what learning method is applicable for what kind of goal violation), and the bias-space search is guided by the performance of the currently held concept description—a learning method is selected based on the violation of performance goals (cf., other work on multistrategy learning (Michalski, 1993)).

Cohen's system, DEXTER (Cohen, et al., 1993), generates machine learning experiments for learning to predict DNA hydration patterns. Based on expert domain knowledge, the system selects subsets of the training examples and subsets of features, with the goal of increasing the accuracy of the learned classifier (the underlying learning system is CART (Breiman, et al., 1984) or Swap-1 (Weiss & Indurkhya, 1991)). DEXTER addresses m_e and s_r. Its policy is to structure its bias space based on domain knowledge about interesting subsets of data and subsets of features. The search is guided by the success with previous biases.

Russell (1986) advocated a framework for building a theory of induction that is similar to the SBS model described below: (i) construct the space of possible classes of higher-level regularities, (ii) search the space for interesting classes, (iii) analyze the results of the search, (iv) apply the results. Russell's space of higher-level regularities comprises those that can be represented in formal logic—the space of all deductive arguments that lead to the base-level rule as their conclusion.

Russell and Grosof have continued this work, looking at inductive bias from the perspective of formal logic (see, e.g., Russell & Grosof (1990)). They consider prior knowledge and learned knowledge as biases for a learning system, in which the hypothesis space is an

intermediate stage between the prior knowledge and the induced theory. They assert that the hypothesis space can be represented as a logical sentence in formal (first-order) logic, which the system derives from what it knows about the domain. The learner's theory of the domain is formed non-monotonically.[6] As a limitation to their declarative approach, they note that "some biases seem to be intrinsically formulated in terms of computational resource use or syntactic structure," which would be difficult to handle with their current approach (Grosof & Russell, 1990, p. 76).

Russell and Grosof address s_r and emphasize theory construction across biases and bias-space search guidance based on learned knowledge. The biases considered are collections of domain knowledge, and the bias-space search guidance takes the form of non-monotonic reasoning. The bias space is structured by the assignment of priorities to default beliefs.

DesJardins' PAGODA (desJardins, 1991; desJardins, 1992) uses probabilistic background knowledge and a model of the system's expected learning performance to compute the expected value of different biases (in this case, sets of features) for different learning goals. Her work uses the bias with the highest expected value for learning, addressing explicitly the tradeoff of accuracy for simplicity of the hypothesis space. Features that increase the expected accuracy marginally but increase the size of the space significantly would not be considered adequately relevant to be included in a bias.

DesJardins' approach addresses s_r. The policy is to structure the bias space based on probabilistic domain knowledge (uniformities) to calculate a bias's expected quality based on expected accuracy and a time-preference function that indicates the degree to which the importance of accuracy on a prediction task changes over time.

3.2. *Summary of Previous Policies*

The similarities and differences among the existing systems for bias selection become apparent when the systems are characterized using the policy techniques presented above. Table 1 summarizes the policies used by previous work, based on these techniques, and includes the SBS Testbed system described in the next section.

Several conclusions can be drawn from the work presented in this section. First, guiding bias selection based on the learned rules is an effective and widely used technique. This can be done based on the set of rules learned in the previous bias (as in STABB), based on a concept description that is constructed as the bias-space search progresses (as in STAGGER), or based on a separate data structure used only by the bias selection system (as in CRL/ISO). Second, it is easier to construct theories across some bias spaces than across others. If the syntax of s_r can vary widely, more effort must be taken to allow a hybrid representation, without which composing knowledge can be very difficult. Even if the syntax of s_r is relatively stable, constructing a theory across biases is easier with some forms of representation than with others. For example, it is not as straightforward to combine multiple decision trees (Buntine, 1991) as it is to combine multiple rule sets (cf., SE-trees, which combine multiple decision trees (Rymon, 1993) and the trees created explicitly for incremental learning (Utgoff, 1989)). A final conclusion from the analysis of existing approaches is that it is possible to structure the bias space, but that to do so there must be prior knowledge regarding the various biases in the space, e.g., knowledge about the priorities of different biases or knowledge about the effects of different biases. If

Table 1. A characterization of existing approaches' methods of searching the bias space.

Bias selection system or approach	Categories of bias addressed	Bias-space search guidance?	Theory construction across biases?	Bias space structuring?
STABB	s_r	yes—based on learned rules	no	no
Other constructive induction	s_r	yes—based on learned rules, data, and/or prior knowledge	yes* & no†	usually not
VBMS	m_r, $\rho_{e,r}$ & $\rho_{r,e}$ (3 learning programs)	no—(potentially) exhaustive search of structured bias space	no	yes—initially random, then learned
CRL/ISO	s_r, $\rho_{e,r}$ & $\rho_{r,e}$	yes—initially randomized, then based on learned rules	no	no
MCS	s_r, $\rho_{e,r}$ & $\rho_{r,e}$	yes—based on heuristic rules based on data and learned rules	yes	yes—global bias
Russell & Grosof	s_r	yes—based on learned rules and prior knowledge (non-monotonic inference)	yes (logical theory of domain)	yes—priorities of default beliefs
PEAK	m_r, $\rho_{e,r}$ & $\rho_{r,e}$ (3 learning programs)	yes—based on violation of goals and structure of bias space	yes (hybrid representation)	yes—effects of methods on goal violations
PREDICTOR	s_r	yes—based on learned rules, assumption definitions and actively selected training examples	yes	possible—heuristics can be ordered based on error penalty & cost of application
adaptive GABIL	m_r	yes—based on learned rules and fitness function (genetic algorithm)	yes	no
DEXTER	m_e & s_r	yes—based on success with previous biases	no	yes
PAGODA	s_r	no	no	yes—probabilistic domain knowledge
SBS Testbed	s_r, s_e, m_r, m_e, $\rho_{e,r}$ & $\rho_{r,e}$	yes—based on learned rules, prior knowledge, and success with previous biases	yes (rule set)	yes—operators explicitly structure bias space

*e.g., STAGGER (Schlimmer, 1987).
†e.g., FRINGE (Pagallo, 1989), CITRE (Matheus, 1989).

such knowledge *is* available, it is important that the approach to bias selection be able to incorporate it.

4. Using an Explicit Policy to Search the Bias Space: The SBS Testbed

Bias selection systems address the problem that in most cases it is not clear *a priori* exactly which set of bias choices to make given the underlying assumptions of the domain. Instead, the assumptions define a space of biases. Just as a bias determines the rule and example spaces and the methods of searching them, the policy defines the bias space (s_b) and the method of searching it (m_b), as shown in Fig. 2.

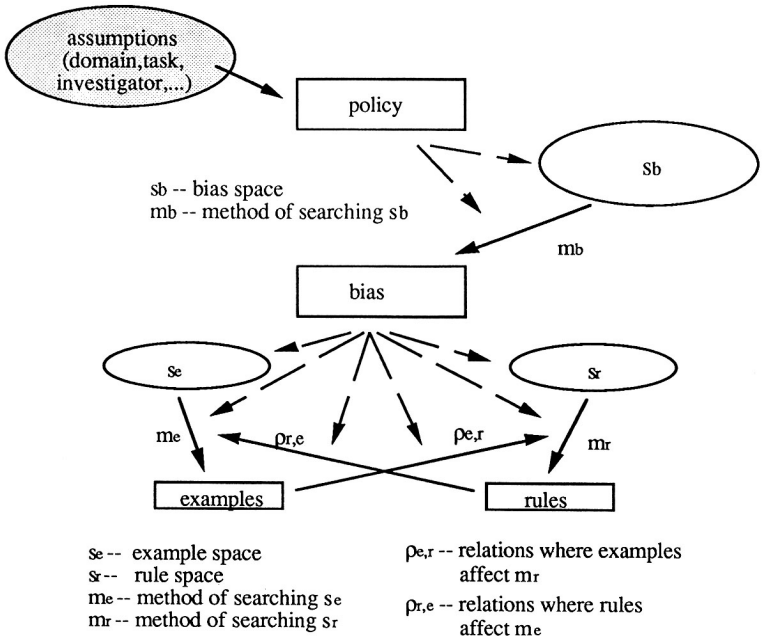

Figure 2. The SBS model: inductive policy defines the bias space and the method of searching the bias space.

4.1. SBS Model

The SBS model (Search of the Bias Space) was developed to facilitate both the study of policies for bias selection and the development of systems that automatically select bias for inductive learning systems. The instantiation of the model used in this paper views biases as states, i.e., as sets of choices of s_r, m_r, s_e, m_e, $\rho_{e,r}$, and $\rho_{r,e}$; *bias transformation operators* are used to move from one bias to the next. The set of bias transformation operators, along with how they are applied and evaluated, define m_b. The dimensions addressed by the operators define s_b. If the underlying learning system represents much of its bias explicitly, the search operators can modify the bias by making different bias choices with respect to this explicit representation. Most often, policies specify how to trade classification accuracy for the reduction of some cost, for example, time, space usage, complexity of the concept description, or costs of using the learned knowledge.

We implemented this model in a system called the SBS Testbed, which allows explicit representation of different inductive policies and experimentation with them. The learning system around which the Testbed has been built is a multiclass version of the RL learning system (Provost, Buchanan, Clearwater, & Lee, 1993; Clearwater & Provost, 1990; Danyluk & Provost, 1993). RL has several convenient explicit "hooks" for changing its bias that can be exploited for these experiments but, as with other learning programs, portions of its bias are implicit. The SBS Testbed allows the specification of policies that address any bias represented explicitly in RL.

A strength of RL as a learning program is that its straightforward search of the rule space allows many different learning biases to be specified. This has been beneficial when using the system as a data-analysis tool, especially when scientists specify particular constraints

Table 2. Characterization of bias for RL. An asterisk indicates a dimension along which a system implemented in the Testbed can select bias, because the dimension is represented explicitly.

Type of bias	Syntactic	Semantic
s_r	disjunctive sets of conjunctive rules, based on attribute/value representation; complexity limit on rules*	set of terms* and value hierarchies* chosen for particular domain; semantic constraints on rule formation (not implemented in Testbed version)
m_r	beam search (width*)	rules* as prior knowledge to bias search
$\rho_{e,r}$	beam search evaluation function*; inductive strengthening* limits search of rule space based on current set of examples	positive and negative performance thresholds* (beam search evaluation function*)
s_e	attribute/value representation	set of terms* chosen for particular domain
m_e	example selection method* (typically manual)	typically none; examples can be weighted based on importance (not implemented in Testbed version)
$\rho_{r,e}$	inductive strengthening* limits examples used in search of rule space based on rules learned so far	typically none

on the rules they want to see (Provost, et al., 1993). Table 2 summarizes RL's bias based on the model presented earlier, further dividing the categories of bias into syntactic or semantic biases, based on the amount of domain knowledge that supports their choice. Asterisks (*) denote those dimensions that are represented in the Testbed. It should be noted that the Testbed can select bias with respect to each of the six categories.

4.2. SBS Testbed Architecture

The user supplies a set of bias transformation operators and an evaluation function to form a specific bias selection system. The control cycle for the SBS Testbed is (see Fig. 3):

(1) form candidate biases with each of the bias transformation operators,
(2) run the learning system with each candidate bias,
(3) compare the biases based on the evaluation function, which usually will factor in the performance of the rules learned with each bias,
(4) choose the bias and/or the corresponding set of rules that performs best with respect to the evaluation function, and
(5) iterate until some stopping criteria are satisfied.

The rules are judged on a set of *evaluation examples*, typically separate from the training examples or the final test set (cf., recent work by Brodley (1993), Quinlan (1993), and Schaffer (1993), who use the performance of learned concept descriptions to pick the best bias). The default evaluation function is classification accuracy; however, other evaluation functions may use domain-dependent criteria for the learned concept description or specific

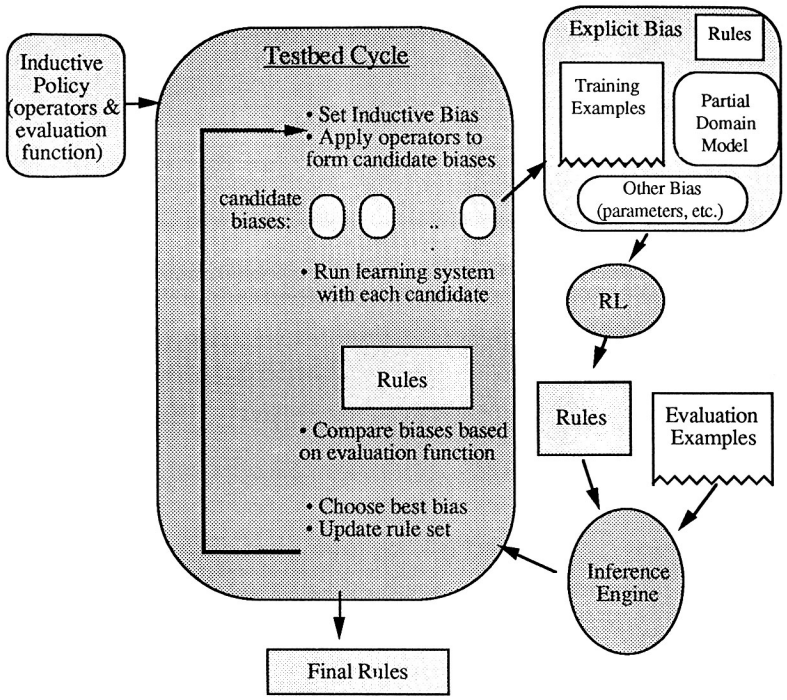

Figure 3. The operator application cycle of the SBS Testbed. The partial domain model describes the domain; it includes attributes, values, value hierarchies, domain-knowledge constraints, etc.

aspects of its performance (e.g., sensitivity, specificity, positive/negative predictive value, costs of errors made).

We built the SBS Testbed to be flexible with respect to the different policies that can be represented. Policies can address any or all of the six categories of bias defined above. In addition, the Testbed policies can structure the bias space explicitly, guide bias space search, and construct concept descriptions from knowledge learned with multiple biases.

4.3. Policy Techniques in the Testbed

As discussed above, a bias space can be structured through restrictions and orderings. This structure can be implemented in the Testbed through (i) the dimensions addressed by the bias selection operators, (ii) the order in which biases are selected by the bias selection operators, and (iii) operator preconditions, which specify the context under which operators are applied.

In the Testbed, bias-space search guidance is implemented through the operators that represent the policy. As in previous systems, guidance can be based on the rules learned in previous biases, the rules' performance on the evaluation examples, or prior domain knowledge. The search of the bias space can be guided in two ways. First, operators can select several different biases, and pick the new bias that performs best. Thus, the system can do hill-climbing (with random walking) under the assumption that a high-performing

bias is more likely to be found near another high-performing bias than near a low-performing bias. A second method for guiding the search of the bias space is to analyze the learned concept description and use the results of the analysis to help suggest subsequent biases.

Concept descriptions can be constructed across multiple biases with two techniques, which can be called *monotonic* and *non-monotonic theory construction*. The former takes the union of the rule set learned with the current bias and the Testbed's current-best rule set, if this union increases the performance. This allows knowledge learned in different biases to be combined. The latter takes the union and then tries to find a good-performing subset via a greedy pruning heuristic (similar to that used by Quinlan (1987)). For convenience, these techniques are provided as options in the Testbed, which can be switched on or off. Alternative theory construction techniques can be instantiated as operators or post- learning actions that modify the currently held rule set in light of newly learned rule sets.

The Testbed also allows the restriction of subsequent rule-space search through the commonly used heuristic: *only consider rules that cover at least one example not covered previously*, which we call *inductive strengthening* because it strengthens subsequent biases based on inductively learned rules. It has been shown that inductive strengthening can reduce the size of the concept description as well as the number of nodes searched when a rule set is provided as prior knowledge to the learner (Provost & Buchanan, 1992b). Testbed operators can take advantage of inductive strengthening by providing a set of rules to RL as prior knowledge. In RL, by default, inductive strengthening takes place based on the currently held concept description, but it can be turned off if policy considerations demand a highly redundant concept description (Provost & Buchanan, 1992a).

5. Experiments

We present four sets of experiments demonstrating the flexibility of the SBS Testbed to represent policies dealing with important tradeoffs in learning regarding time, space, accuracy, and cost. The experiments test two main claims, one general and one specific. First, the SBS Testbed is flexible in that it can represent policies that deal with a variety of types of bias; previous systems were very limited in the different types of bias that could be addressed (see Table 1). In particular, we consider parameter selection, term selection, and example selection, based on different tradeoffs. Our second claim is that when the SBS Testbed is given policies to search the bias space automatically, it finds biases that satisfy the tradeoffs and thereby allow effective learning.[7]

Our purpose is not to compare any specific policy with previous policies, but to show that by representing inductive policy explicitly it is possible to implement a variety of policies in a single system. We believe that it is important not to have to build a new learning system whenever a new pragmatic concern is encountered. The SBS framework provides a method for taking pragmatic considerations as input to the system, in the form of inductive policies. The first three sets of experiments demonstrate the flexibility of the Testbed to address different dimensions in the bias space, through the specification of different operators. The fourth set of experiments shows that by changing the system's evaluation function based on different desired tradeoffs, different behavior can be elicited even from the same training data.

In addition, we provide further evidence, augmenting the analysis of previous work, that very similar inductive policy techniques are applicable for bias selection along very

different bias dimensions. We show that the SBS framework allows expressions of simple policies that can guide bias-space search effectively. One could imagine implementing more complex policies for searching the bias space (e.g., simulated annealing).

5.1. Experiment Set #1: Parameter Selection

To show how a policy can favor accuracy over learning time we consider the selection of parameters for RL. We show that the policy is effective by showing that the rules learned had comparable or better accuracy than rules learned by manually setting RL's bias or by using a decision-tree learning program.

Consider four parameters: (a) the beam width of the beam search, (b) the maximum number of conjuncts in any of the rules considered, (c) a limit on the fraction of the positive examples a rule must cover and (d) a limit on the fraction of the negative examples a rule may cover, e.g., to allow for noisy data. For any particular application in a given domain, it is not clear *a priori* how these search parameters for RL's covering procedure should be set. For example, for the well-known problem of learning to classify mushrooms as edible or poisonous (Schlimmer, 1987), the size of the syntactically defined space of conjunctive descriptions is greater than 10^{15} using 22 multi-valued attributes. Thus it is important to select a bias that restricts search, while still allowing the learning program to find a satisfactory concept description.

To show that we can trade increased learning time for increased accuracy, we implemented a randomized search of the bias space with the objective of maximizing classification accuracy. The bias-space search terminated when 100,000 partial rules had been searched by RL since the last improvement in accuracy. This policy was implemented in the Testbed with an operator (see Fig. 4) that randomly selects a positive threshold, a negative threshold, a rule complexity, and a beam width, and with an operator that tested for convergence.

We ran the randomized system with three sets of data from the UCI Repository: the mushroom domain, the automobile domain, and the heart disease domain. The first domain was chosen because it was of interest to the authors; the latter two because they contained both symbolic and numeric data. The results are summarized in Table 3, which lists the domain, the numbers of training and test examples, and the test-set predictive accuracy (averaged over ten random training/test partitions) of rules learned by three different systems: (i) the Testbed system using the operator of Fig. 4, (ii) RL with manual bias selection (by the first author), and (iii) RL-DT (a decision tree program based on C4 (Quinlan, 1986b) that accompanies RL). The decision trees were converted to rules, which were then pruned based on the techniques of Quinlan (1987). All classifications were made by an inference

Preconditions:
<none>
Actions:
Select positive threshold randomly from (0.01, 1.0)
Select negative threshold randomly from (0, positive threshold)
Select rule complexity randomly from (1, total # of terms)
Select beam width randomly from (1, 10,000)

Figure 4. Testbed operator for randomly selecting bias along four dimensions: the positive and negative performance thresholds, the beam width, and the maximum conjunct limit.

Table 3. Results for three learning problems where the bias space was searched under different policies (averages over 10 trials with 95% confidence intervals using Student's t distribution are shown).

Domain (UCI repository)	Number of examples (train/test)	Average test accuracy (%) (with 95% confidence interval) System-(i) (randomized selection)	System-(ii) (manual selection)	System-(iii) (fixed bias: decision tree)
Automobiles (3 price classes)	(133/66)	84.5 ± 4.5	80.4 ± 1.7	78.2 ± 1.9
Heart disease	(200/103)	78.0 ± 2.3	76.9 ± 1.3	73.4 ± 2.2
Mushrooms	(100/915)	96.3 ± 1.5	95.0 ± 1.1	95.1 ± 1.8

engine that used a voting strategy when multiple rules predict different classes. The results with the decision-tree rules are included for comparison with a typical fixed-bias system. All systems were trained on the same number of examples. System-(i) requires a separate set of examples for bias evaluation; to facilitate comparison we reused the training examples as the evaluation examples (these were not the final test data).

These experiments show that a parameter selection policy dealing with one type of tradeoff can be represented in the SBS Testbed, and the Testbed can find biases to learn satisfactory concept descriptions with respect to that policy. The tradeoff involved here is a typical one in machine learning: higher accuracy for increased search time. For example, system-(i) took longer than any single run of RL, but manual bias selection (with system-(ii)) involved several RL runs to find a high-accuracy bias. System-(i) exhibits significantly higher accuracy in the Auto and Heart domains because it had more patience than the author did in waiting for convergence. Note that a randomized policy alone works well in these domains mainly because there are many sufficient biases (the concepts are relatively easy to learn, cf. Holte (1993)).

To provide further evidence that increased time can be traded effectively for increased accuracy, we ran system-(i) as an "anytime" learner: the system was stopped at various points during its search, and the accuracy of its current concept description was measured. If the quality of the concept descriptions learned with different biases varies, the quality of the concept descriptions system-(i) learns will increase monotonically (modulo errors in the evaluation of the concept descriptions' qualities) until the "best" concept description is learned.

Figure 5 summarizes the results of 90 runs on the mushroom data. We chose landmarks of 100, 200, 400, ..., 25600 RL search nodes, and provided system-(i) with a termination operator that stopped the bias-space search after the underlying system had completed the run where the landmark was exceeded. The best set of rules at this point was used as the concept description for this run. The runs were performed in a manner similar to those above, with 10 runs per landmark. The figure plots the average classification accuracy of the resultant concept descriptions versus the average number of actual nodes searched for each landmark; 95% confidence intervals are shown. Note that in addition to the marked increase in accuracy up to 1000 search nodes, the figure also shows that the accuracies become more consistent as the system performs more bias-space search.

In principle, the randomized policy scales the amount of search of the bias space to the amount of time available, from a random selection at one extreme to an exhaustive search (effectively) at the other. Here the only non-negligible factor in the Testbed's overhead

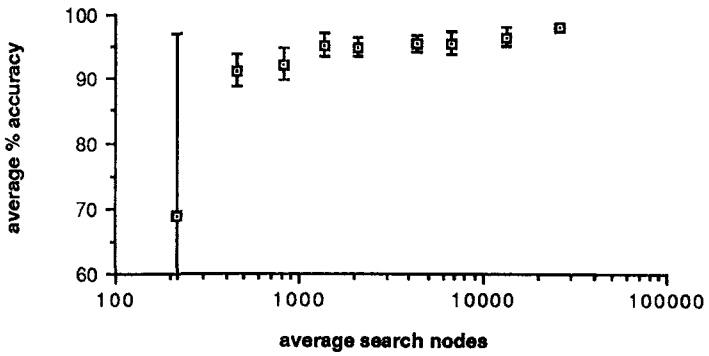

Figure 5. System-(i)'s performance in the mushroom domain increases with the amount of time spent searching the bias space.

is the bias evaluation. However, it is important to note that the randomized policy is a completely uninformed bias-space search; it uses none of the three techniques described above to increase its efficiency in searching the bias space. The next section describes experiments where a randomized policy alone is not effective, but an augmented policy is.

5.2. *Experiment Set #2: Term Selection*

In order to investigate policies for a different tradeoff and a different type of bias, we chose to address the following pragmatic consideration: if there are very many terms that might be potentially relevant and processing capability is limited, e.g., there is not enough space to learn with all the terms simultaneously, one may need to limit the set of terms used for any learning run. If the number of irrelevant terms is large, and no information is provided as to probable relevance, the bias space formed by taking small subsets of the large set will be too sparse for a randomized search to be effective and tractable. However, we show that by utilizing the techniques outlined in Sections 2 and 4, a randomized search can be enhanced so that it overcomes the problem of a sparse space and can be effective in selecting a good bias. The policies given to the Testbed were: randomized search of small sets of terms, randomized search guided by learned relevance knowledge, and randomized searches where the concept descriptions are constructed across biases using different methods of theory construction.

Note that this is a slightly different problem from that addressed in previous work on term selection (e.g., Littlestone (1988)). First, in the present work the pragmatic assumption is that it is not possible to learn with all the terms simultaneously; previous work, on the other hand, tries to reduce the number of mistakes made when learning with a large number of terms. Second, it is important to avoid building a new system for every new pragmatic consideration encountered. This section shows that by giving the Testbed an appropriate policy, the system can deal effectively with a new pragmatic consideration. Almuallim and Dietterich (1991) have studied the problem of learning with many irrelevant features using an exhaustive-search policy and conclude that irrelevant features should be removed from the training data before popular inductive learning algorithms (viz., ID3 and the like) can learn well. The problem of relevance of terms also has been studied by several other

Table 4. Target concept for synthetic learning task. TP/P and FP/N denote the (fractional) true positive and false positive coverages of each rule in the target concept. Each (Xx_i) is a feature: (attribute value).

Rule	TP/P	FP/N
$(S\,s_1)(T\,t_1) \rightarrow C1$	0.8	0.03
$(L\,l_1)(M\,m_1)(N\,n_1) \rightarrow C1$	0.4	0.02
$(I\,i_1)(J\,j_1) \rightarrow C1$	0.5	0.03
$(D\,d_1)(E\,e_1) \rightarrow C1$	0.6	0
$(A\,a_1) \rightarrow C1$	0.4	0
$(R\,r_1) \rightarrow C2$	0.7	0.01
$(O\,o_1)(P\,p_1)(Q\,q_1) \rightarrow C2$	0.5	0.04
$(K\,k_1) \rightarrow C2$	0.6	0.04
$(G\,g_1)(H\,h_1) \rightarrow C2$	0.3	0.01
$(B\,b_1)(C\,c_1) \rightarrow C2$	0.7	0

AI researchers (Samuel, 1963; Subramanian, 1989; Riddle, 1989; Gordon, 1990; Kira & Rendell, 1992; desJardins, 1992), and in the statistics literature (Devijver & Kittler, 1982).

5.2.1. A Synthetic Domain. To simulate the problem of having a very large set of terms most of which are irrelevant, we used a synthetic domain. The synthetic domain has 22 attributes, each with between 2 and 12 nominal values. The data were generated from a set of 10 rules, which are depicted in Table 4. Some of the rules are "heuristic" rules, in that they have non-null false positive coverage. The example generator gave a set of data comprising examples of concepts $C1$ and $C2$, such that the constraints given by the rules and their coverages were satisfied. We chose this synthetic domain because we could specify a concept that is relatively complex, but for which we know a *priori* how to set RL's parameters to regenerate the rules that generated the data. Thus we can concentrate on selecting bias with respect to the set of terms used. To this set of 22 attributes we added 198 new Boolean attributes, giving a total of 220 attributes. The example generator filled in the irrelevant attributes with random values. See Provost (1992b) for more details.

5.2.2. Term Selection Experiments. Using data generated with the augmented attribute set, we ran several experiments with different term selection policies. The experiments show that bias selection policies can be represented in the Testbed to deal effectively with a second type of bias and a second tradeoff: increasing learning time for a reduction in space usage to learn a satisfactory concept description. Furthermore, the experiments illustrate the utility of the inductive policy techniques described in Sections 2 and 4.

The results of these experiments are summarized in Table 5. For each experiment, the table lists three quantities averaged over 10 runs with different randomly generated training sets of 200 examples. The quantities compared in the table are the total number of nodes searched in the underlying rule-space searches, the number of rules in the learned rule set, and the classification accuracy on a separate test set of 1000 examples.

Experiment 0 shows the results of running RL with the original 22 attributes, known to be relevant; in Experiment 1 we used the entire set of 220 attributes. These runs generated figures for comparison. RL's bias along the dimensions considered in the previous section was held constant at the strongest values that would allow the system to learn the set of rules from which the data were generated. The results of Experiment 0 are as would be

Table 5. Experiments with term selection policies (averages over 10 trials with 95% confidence intervals using Student's t distribution are shown).

Experiment & system no.	Experiment	Nodes searched	Rules learned	Accuracy
0	RL with 22 attributes	535 ± 88	26 ± 3	96.9 ± 0.6
1	RL with 220 attributes	6825 ± 195	842.1 ± 174.1	90.4 ± 1.3
2	Random subsets of 22 attributes	19636 ± 3035	15.8 ± 6.2	76.3 ± 2.7
3	Selection of approx. 22 attributes based on learned relevance knowledge	26804 ± 5174	68.6 ± 10.5	82.8 ± 2.0
4	System-2 + monotonic theory construction across biases	27152 ± 5292	120.8 ± 42.9	91.5 ± 1.6
5	System-4 + inductive strengthening	14930 ± 3368	42.3 ± 12.4	91.9 ± 1.4
6	System-5 with non-monotonic theory construction	25913 ± 5781	12.1 ± 1.0	95.4 ± 1.3

Preconditions:
<none>
Actions:
Select 22 attributes randomly from the large set (220 attributes)

Figure 6. Testbed operator for randomly selecting a subset of terms (System-2).

expected given that a good bias is known in advance—high accuracy and short search time. In Experiment 1, over 800 rules were learned in a search of approximately 6800 nodes. The large number of rules learned is due to the existence of such a large number of randomly generated Boolean features. Given a rule that almost meets the thresholds, in many cases there will be one or more randomly generated features that can be added to the rule to meet the thresholds. This fact also accounts for the decrease in classification accuracy as compared to running RL with only the relevant attributes.

Let us now assume that our computational power is limited so that we can learn with only about 22 attributes at a time. We used the SBS Testbed to build systems that select subsets of approximately 22 terms, restricting the otherwise prohibitively expensive bias space. We start with a randomized term selection policy. A system was instantiated in the Testbed with a single operator that chose a random subset of 22 of the 220 terms. This operator is depicted in Fig. 6; we will refer to this system as "System-2," based on the experiment number listed in Table 5, and the systems for the subsequent experiments similarly. The convergence criterion was 10,000 nodes searched without an improvement in score; the score was classification accuracy on a separate set of 200 evaluation examples. We see that as compared to RL with the larger set of terms, System-2 used more search to learn fewer rules with lower accuracy. This is because with the smaller sets of terms, there is much less freedom to fit the data with random features. However, none of the randomly chosen subsets of terms contained enough of the "correct" terms. Randomized search is not sufficient for a search of this bias space with the convergence criteria given, although given enough time, probabilistically speaking, the randomized search will hit the correct set of attributes eventually.

Preconditions:
<none>
Actions:
Include each attribute a_i in attribute set with probability p_i
(initially uniform: $p_i = \frac{\text{desired \# of attributes}}{\text{total \# of attributes}}$)
Post-Learning Actions:
For each rule learned and attribute a_i mentioned in that rule
if $p_i < 1$ increment p_i by ε, where $\varepsilon = \max(0.01, 1-p_i)$ and decrement each p_j, such that $j \neq i$, and a_j was selected in this round, by $\frac{\varepsilon}{\text{\# of selected attributes - 1}}$

Figure 7. Testbed operator for probabilistically selecting a subset of terms, with guidance based on learned relevance knowledge (System-3).

One technique to extend a simple bias-space search, suggested by the analysis of existing systems, is to use knowledge learned during the search of the bias space to guide the selection of future biases. For the current problem, learned knowledge as to the probable relevance and irrelevance of the attributes can guide the search to subsets of terms that include more of the relevant terms and fewer of the irrelevant terms, in effect, choosing terms that correlate well with the data—both alone and in conjunction with other terms. In Experiment 3, the randomized search was enhanced to take advantage of knowledge learned about the relevance of the attributes. System-3's operator is shown in Fig. 7.

For the current set of attributes, System-3 either selected or rejected each attribute with a probability from a list of attribute/probability pairs. The system updated this data structure based on the results of each run of the learning system. Through this process, attributes that contribute to good rules have an increased probability of appearing in future sets of attributes, while attributes that never appear in good rules have a decreased probability of appearing in future sets. The performance of System-3 shows a significant increase in classification accuracy over System-2 indicating that the search guidance indeed leads the system to better biases. The number of rules learned is increased, due to the larger number of relevant terms. The number of nodes searched is a little larger than it was for Experiment 2, indicating that it took some time to build up the knowledge needed to converge on a (better-performing) bias.

The accuracy achieved in Experiment 3 is still not as high (83%) as that achieved with RL and the large set of terms (90%).[8] Several modifications of the policy yielded similar results. Viewing traces of the evolution of the relevance data structure showed that over many runs, terms achieved high relevance in rough proportion to the simplicity of the rules in which they appear. This indicates that this policy did not deal well with feature interaction. Previous work has shown that similar, but simpler, policies that try to build sets of terms incrementally also have problems with feature interaction, e.g., sequential forward selection (Devijver & Kittler, 1982; Kira & Rendell, 1992).

Experiment 4 used System-2 enhanced with monotonic theory construction across biases. As described in Section 4.3, this technique simply replaces the current "best" rule set with the union of this rule set and the rule set learned with the new bias, if the combined set shows a performance improvement. By the results shown in Table 5, we can see that the latter

system performed better in terms of classification accuracy, while searching approximately the same number of nodes.

The concept descriptions learned by System-4 were more complex than those learned by either System-2 or System-3. In Experiment 5, System-4 was run with the inductive strengthening heuristic turned on and the Testbed's current rule set used as prior knowledge for subsequent runs (see Section 4.3). We see a decrease by almost half in the amount of search needed as well as a decrease in the complexity of the resultant concept description.

The final experiment listed in Table 5 shows the effect of using the non-monotonic method of theory construction (see Section 4.3) in a system in all other respects equivalent to System-4. The result is that by trading off time spent learning and constructing the theory, a small set of rules can be learned that performs very well even given the space constraints.

These results provide support for our two main claims, showing that a system based on the SBS model is flexible in that it facilitates experimentation with different inductive policies for term selection, leading to an effective policy. Additionally, the results indicate that a simple method for using knowledge learned during bias selection to construct concept descriptions across multiple biases was more effective than a simple method for using learned knowledge to guide the bias-space search. This is interesting because while the policies of most previous bias selection systems implemented some form of bias-space search guidance, only about half of these systems constructed concept descriptions from knowledge learned in multiple biases (see Table 1).

5.3. *Experiment Set #3: Example Selection*

To demonstrate further the flexibility of the SBS Testbed, we experimented with representing a policy for example selection in the form of incremental batch learning (Clearwater, et al., 1989). The tradeoff of interest in this experiment was to allow somewhat decreased accuracy (from what would be achievable with a batch learner) in order to be able to learn from many examples under space constraints. Implementing such a policy demonstrates how inductive policies can deal with aspects of inductive bias relating to the selection of examples (m_e) within the same framework, and using similar techniques, as more conventional aspects of inductive bias.

In essence, the policy is: randomly divide a large set of examples into small subsets; learn a set of rules with the first subset as the training set; switch to the next subset; filter the rules learned with the previous subset using the new training set; and then learn with the new subset, constructing the overall theory incrementally with subsequent learning addressing incompleteness in the currently held theory. The operator for this incremental batch learning policy is shown in Fig. 8.

Preconditions:
<none>
Actions:
load next example set (partitioned as preprocess)
update coverages of current rule set
remove rules that do not satisfy thresholds
invoke inductive strengthening and learn with new rule set

Figure 8. Testbed operator for incremental batch learning.

Table 6. Results of batch learning and incremental batch learning policy on NYNEX MAX learning task (space constraints limited batches to 500 examples).

# Training examples per run	# Test examples	Average% accuracy	Standard deviation	Number of runs
500	2457	86.5	1.0	10
10 batches of 500	845	90.7	0.9	10

As shown in Table 6 the simple incremental batch learning policy is sufficient for learning a satisfactory theory on the NYNEX MAX learning task (Rabinowitz, Flamholz, Wolin, & Euchner, 1991; Danyluk & Provost, 1993) on a platform where space constraints previously had prohibited satisfactory learning (Provost, 1992b). For this task, satisfactory learning had been defined beforehand by the domain experts as greater than 90% accuracy. These results provide further support for our first claim that the SBS Testbed system is flexible—a policy can be represented for example selection within the same framework as parameter selection and term selection. They also provide support for our second claim—the incremental batch learning policy was effective for learning a satisfactory rule set under space constraints. Interestingly, as in the previous section, theory construction across biases alone is sufficient for effective learning (no structuring or search guidance is used here).

5.4. Experiment Set #4: Trading off Accuracy for Reduced Error Cost

In previous sections we experimented with different bias selection operators in order to demonstrate the flexibility of the SBS Testbed. In this section we further demonstrate the Testbed's flexibility by experimenting with the bias evaluation function. The results show that policies can be represented in the Testbed that select bias effectively with respect to different desired tradeoffs involving accuracy and the cost of errors.

In the mushroom domain the cost of making a mistake is asymmetric. Under normal circumstances, no harm is done when an edible mushroom is classified as poisonous. In contrast, classifying a poisonous mushroom as edible can have mortal consequences. In this domain, the assumption that one prediction is more costly than another should lead to a different inductive policy from that taken when all mistakes can be weighted equally. Obviously, a completely safe policy would be to predict every mushroom is poisonous—dangerous predictions would never be made. However, this approach would never allow any mushroom to be eaten.

For these results we used the ClimBS system, implemented in the SBS Testbed. As described in Provost (1992a), ClimBS is similar to the randomized system of Section 5.1, but performs a hill-climbing search in RL's bias space and uses nonmonotonic theory construction.

Table 7 shows the results of several experiments in which ClimBS was used to learn rules for the mushroom domain. The bias evaluation function was varied across the experiments to reflect different assumptions about the tradeoff of accuracy versus the cost of making risky errors. The table lists the number of rules learned, the predictive accuracy of the rule set, and the fraction of the predictions that might be risky. The results are averaged over 10 trials, 95% confidence intervals are given; 100 randomly selected examples were used for learning, 400 for bias evaluation, and 515 for testing; we chose a small training set in

Table 7. Experiments I–IV compare the accuracy and risk (percentage of predictions that are possibly fatal) of learning with different inductive policies represented as bias evaluation functions in ClimBS (and are described in the text). Results are averages over 10 runs; 95% confidence intervals are given. Training sets contained 100 examples randomly chosen from 1015; 400 randomly chosen examples were used for bias evaluation, and the remaining 515 examples were used for testing.

Experiment	No. rules	Accuracy	% Risk
I	8 ± 2	96.3 ± 1.3	2.7 ± 2.0
II	9 ± 1	95.2 ± 0.9	0.87 ± 0.48
III	8 ± 2	91.5 ± 5.2	0.74 ± 0.45
IV	6 ± 2	82.2 ± 14.8	0.16 ± 0.28

order to study error cost. These 1015 examples comprise every eighth mushroom in the database and can be shown to be a representative sample—ClimBS can learn a set of rules with these 1015 examples that can predict the edibility of the remaining 7000+ examples with an accuracy of 99.7% on the remaining examples in the database.

In Experiment I, the default evaluation function (only consider predictive accuracy, use voting to apply rules) was used. Although the classification accuracy is impressive, over two percent of the examples were dangerously classified as edible, when they were actually poisonous. In Experiment II the evaluation function compared biases based on predictive accuracy, applying rules with a "better safe than sorry" evidence-gathering strategy: call a mushroom poisonous if any rule says it is poisonous. The accuracy of the resultant descriptions are not degraded much, while the fraction of risky predictions is reduced to one-third its previous value.

Experiments III and IV use a linear combination of predictive accuracy and number of risky predictions to evaluate the rule sets learned with different biases. In particular, the function used was: f = number of correct predictions $- w$ * number of risky predictions. In Experiment III, $w = 10$; in Experiment IV, $w = 50$. One can see the tradeoff of classification accuracy for safe predictions that is manifested in the rule sets learned by ClimBS.

The SBS Testbed allows for the specification of the relative value of different rule sets with respect to their performance. This explicit representation of how the system should deal with the accuracy/safety tradeoff allows the system to select biases to learn a concept description that gives a good score with respect to this function, and thereby performs well with respect to the tradeoff. This work on cost-sensitive learning is extended by Provost (1994), where a similar policy is shown to be effective for trading off accuracy for a reduction in monetary costs when diagnosing errors in the telephone local loop.

6. Conclusions and Future Directions

We have presented a model of six categories of bias that define a bias space, and have augmented it with a separate concept of inductive policy, which takes account of the pragmatics of learning, i.e., the tradeoffs that are made with respect to goals defined in each domain. A future avenue for research is build a system like the SBS Testbed, where policies are represented in a less procedural fashion. As a first step in this direction, this work begins to identify a taxonomy of pragmatic concerns for learners. At a high level, pragmatic concerns can be divided based on whether they apply to the learner itself or the learned concept

description. Finer grained concerns address benefits (accuracy, understandability), costs (of measuring features, of making mistakes), and resource usage (time, space).

The major drawback to this search-based approach to bias selection is computational efficiency. Learning with a large number of different biases can be very expensive in terms of run time. As discussed in Section 5.1, tradeoffs can be made between the amount of search time and other desiderata (viz., high accuracy); however, addressing the problem of scaling up to larger problems becomes especially important when a learner is trying a large number of different biases. Machine learning research has begun to address the issue of scaling up learning programs to very large problems (Catlett, 1992; Musick et al., 1993; Wirth & Catlett, 1988; Provost & Aronis, in press; Provost & Hennessy, 1994). Scaling up using parallelism is especially relevant to bias selection. Using massively parallel architectures, the time needed for an individual run of a learning system can be reduced by two to three orders of magnitude, enabling bias selection for larger problems (Provost & Aronis, in press). Alternatively, by taking advantage of a network of workstations, one can scale up by learning with different biases on different machines and use cooperation to construct a satisfactory concept description (cf. work on distributed machine learning (Chan & Stolfo, 1993; Provost & Hennessy, 1994)).

In order to avoid significant recoding in building a new system for each new learning task, learning systems must be flexible. We have shown that by modeling bias selection as search, a flexible system can be built that can learn differently under different pragmatic constraints. We have shown that within the same framework, the SBS Testbed can perform parameter selection, term selection, and example selection. In addition, we have shown that the Testbed can learn different concept descriptions based on different tradeoffs between accuracy and the cost of errors. The Testbed achieves this flexibility by explicitly representing an *inductive policy* as input—i.e., a strategy for selecting bias that satisfies a task's pragmatic constraints.

In the SBS Testbed, inductive policies are represented as a set of bias selection operators, and a function for evaluating biases. Thus, it is possible to implement policies for selecting bias along any dimension that is represented explicitly in the underlying learning system. This suggests that an important future direction for inductive bias research is to provide learning systems that are more flexible in the different kinds of biases they can employ. For example, a limitation of the SBS Testbed is that it currently uses only RL as the underlying learning system; as Section 3 shows, much work has been done on policies for selecting bias with respect to the description language beyond what is available in RL. The Testbed could be augmented with many underlying learners. More complex techniques would have to be developed for bias-space search guidance, bias-space structuring, and the construction of theories from multiple biases.

In addition, we have shown from an analysis of previous bias selection systems and new results that the three main techniques for building non-trivial policies, structuring the bias space, using learned knowledge to guide bias selection, and constructing a theory across multiple biases are not specific to any particular dimension in the bias space. Previous systems that address very different dimensions use similar techniques. Instantiations of these techniques in the Testbed have been shown to be effective across a variety of bias-space dimensions. The experiments in Section 5.2 suggest that for at least one task, constructing a concept description from knowledge learned in multiple biases is more effective than trying to guide search to a satisfactory bias. We would like to see if this is a general phenomenon, which would suggest that bias selection systems should concentrate on techniques for combining knowledge learned in multiple biases (cf., Buntine, 1991).

Acknowledgments

This work benefited from discussions (some many years ago) with C. West Churchman, Scott Clearwater, Andrea Danyluk, Edward A. Feigenbaum, Randy Jones, Jonathan J. King, Joshua Lederberg, Tom Mitchell, Jeff Schlimmer, Kurt VanLehn, and the Pitt machine learning discussion group and from reviews by anonymous referees. We would like to thank Jeff Schlimmer for supplying a version of C4 (which became RL-DT), the UCI Repository for access to the machine learning database, and Diana Gordon, Joel Martin and Ron Rymon for comments on drafts. Heart disease data were supplied by V.A. Medical Center, Long Beach and Cleveland Clinic Foundation: Robert Detrano, MD, Ph.D. This work was supported by NLM grant 1-R01-LM05104, NYNEX Science and Technology, Inc., an IBM Graduate Fellowship, and the W.M. Keck Center for Computational Biology.

Notes

1. Any procedure for selecting a bias constitutes a bias—but at a different level. This is the point of separating policy considerations that include criteria for evaluating a bias from the inductive bias that directly guides the learning, and the point of allowing different *policies* to be represented.
2. However, often search methods are designed for particular representations, so in practice changing only one may be less than straightforward.
3. Note how this corresponds to the notions of preference and restriction biases. Once a particular inductive policy is implemented, it becomes a meta-level bias to the learning system.
4. Automatically changing the framework within which a learning program operates was discussed explicitly as early as 1978 (Buchanan, et al., 1978).
5. A bias is "Pareto optimal" with respect to a set of criteria if there is no bias that will improve performance with respect to one criterion without degrading it with respect to another.
6. Russell and Grosof employ a non-monotonic logic, based on circumscription (see Lifschitz, 1987; McCarthy, 1980), that allows one to express prioritized defaults directly (Grosof & Russell, 1990).
7. "Effective" learning means different things in different pragmatic situations. In one context, one may want to spend considerable time maximizing accuracy. In another context, one may be satisfied with lower accuracy in exchange for timely results.
8. It does provide a simpler rule set.

References

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

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees*, Belmont, CA: Wadsworth International Group.

Brodley, C.E. (1993). Addressing the selective superiority problem: automatic algorithm/model class selection. *Proceedings of the Tenth International Conference on Machine Learning* (pp. 17–24), San Mateo, CA: Morgan Kaufmann.

Brodley, C.E. (this volume). Recursive automatic bias selection for classifier construction.

Buchanan, B., Johnson, C., Mitchell, T., & Smith, R. (1978). Models of learning systems, in J. Beltzer (Ed.), *Encyclopedia of computer science and technology* (pp. 24–51).

Buntine, W. (1991). *A theory of learning classification rules*, Ph.D. Thesis, University of Technology, Sidney.

Catlett, J. (1992). Peepholing: choosing attributes efficiently for megainduction, in D. Sleeman & P. Edwards (Eds.), *Proceedings of the Ninth International Conference on Machine Learning* (pp. 49–54), San Mateo, CA: Morgan Kaufmann.

Chan, P., & Stolfo, S. (1993). Toward parallel and distributed learning by meta-learning. *Proceedings of the AAAI-93 Workshop on Knowledge Discovery in Databases.*

Clearwater, S., Cheng, T., Hirsh, H., & Buchanan, B. (1989). Incremental batch learning. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 366–370), San Mateo, CA: Morgan Kaufmann.

Clearwater, S., & Provost, F. (1990). RL4: a tool for knowledge-based induction. *Proceedings of the Second International IEEE Conference on Tools for Artificial Intelligence* (pp. 24–30), Los Alamitos, CA: IEEE C.S. Press.

Cohen, D., Kulikowski, C., & Berman, H. (1993). Knowledge-based generation of machine learning experiments: learning with DNA crystallography data. *Proceedings of the First International Conference on Intelligent Systems for Molecular Biology.* Menlo Park, CA: AAAI Press.

Danyluk, A.P., & Provost, F.J. (1993). Small disjuncts in action: learning to diagnose errors in the telephone network local loop. *Proceedings of the Tenth International Conference on Machine Learning (ML-93)* (pp. 81–88), San Mateo, CA: Morgan Kaufmann.

DeJong, G., & Mooney, R. (1986). Explanation-based learning: an alternative view, *Machine Learning*, 1:287–316.

desJardins, M. (1991). Probabilistic evaluation of bias for learning systems. *Proceedings of the Eighth International Workshop on Machine Learning* (pp. 495–499), San Mateo, CA: Morgan Kaufmann.

desJardins, M. (1992). *PAGODA: a model for autonomous learning in probabilistic domains*, Ph.D. Thesis, University of California at Berkeley (available as Technical Report UCB/CSD 92/678).

Devijver, P.A., & Kittler, J. (1982). *Pattern recognition: a statistical approach*, Prentice Hall.

Dietterich, T. (1991). Machine learning: issues, answers, and quandaries, Keynote Lecture, AAAI-91.

Dietterich, T., & Michalski, R. (1983). A comparative review of selected methods for learning from examples, in R. Michalski, J. Carbonell, & T. Mitchell (Eds.), *Machine learning: an artificial intelligence approach* (pp. 41–81), Palo Alto, CA: Tioga.

Gordon, D. (1990). *Active bias adjustment for incremental, supervised concept learning*, Ph.D. Thesis, Department of Computer Science, University of Maryland.

Grosof, B., & Russell, S. (1990). Shift of bias as non-monotonic reasoning, in Brazdil & Konolige (Eds.), *Machine Learning, meta-reasoning, and logics*, Boston, MA: Kluwer Academic Publishers.

Haussler, D. (1988). Quantifying inductive bias: AI learning algorithms and Valiant's learning framework, *Artificial Intelligence*, 36:177–221.

Holder, L. (1990). The general utility problem in machine learning. *Proceedings of the Seventh International Conference on Machine Learning* (pp. 402–410), San Mateo, CA: Morgan Kaufmann.

Holte, R.C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11(1):63–90.

Kira, K., & Rendell, L. (1992). The feature selection problem: traditional methods and a new algorithm. *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 129–134), Menlo Park, CA: AAAI Press.

Korf, R.E. (1985). Depth-first iterative deepening: an optimal admissible tree search, *Artificial Intelligence*, 27:97–109.

Lifschitz, V. (1987). Circumscriptive theories: a logic-based framework for knowledge. *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 364–368), San Mateo, CA: Morgan Kaufmann.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm, *Machine Learning*, 2(4):285–318.

Matheus, C. (1989). *Feature construction: an analytic framework and an application to decision trees*, Ph.D. Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign.

McCarthy, J. (1980). Circumscription—a form of non-monotonic reasoning, *Artificial Intelligence*, 13:27–39.

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

Michalski, R.S. (1993). Special issue on multistrategy learning, *Machine Learning*, 11(2/3).

Mitchell, T. (1978). *Version spaces: an approach to concept learning*, Ph.D. Thesis, Department of Electrical Engineering, Stanford University.

Mitchell, T. (1980). *The need for biases in learning generalizations* (Technical Report CBM-TR-117), Department of Computer Science, Rutgers University.

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

Mitchell, T., Utgoff, P., & Banerji, R. (1983). Learning by experimentation: acquiring and refining problem-solving

heuristics, in R. Michalski, J. Carbonell, & T. Mitchell (Eds.), *Machine learning: an artificial intelligence approach* (pp. 163–190), Palo Alto, CA: Tioga.

Musick, R., Catlett, J., & Russell, S. (1993). Decision theoretic subsampling for induction on large databases. *Proceedings of the Tenth International Conference on Machine Learning* (pp. 212–219), San Mateo, CA: Morgan Kaufmann.

Pagallo, G. (1989). Learning DNF by decision trees. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 639–644), San Mateo, CA: Morgan Kaufmann.

Provost, F.J. (1992a). ClimBS: searching the bias space. *Proceedings of the Fourth International Conference on Tools with Artificial Intelligence* (pp. 146–153), Los Alamitos, CA: IEEE Computer Society Press.

Provost, F.J. (1992b). *Policies for the selection of bias in inductive machine learning*, Ph.D. Thesis, Computer Science Department, University of Pittsburgh.

Provost, F.J. (1993). Iterative weakening: optimal and near-optimal policies for the selection of search bias. *Proceedings of the Eleventh National Conference on Artificial Intelligence* (pp. 749–755), Menlo Park, CA: AAAI Press.

Provost, F.J. (1994). Goal-directed inductive learning: trading off accuracy for reduced error cost. *Proceedings of the AAAI Spring Symposium on Goal-Directed Learning.*

Provost, F.J., & Aronis, J. (in press). *Scaling up inductive learning with massive parallelism, Machine Learning.*

Provost, F.J., & Hennessy, D. (1994). Distributed machine learning: scaling up with coarse-grained parallelism. *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology.* Menlo Park, CA: AAAI Press.

Provost, F., & Buchanan, B. (1992a). Inductive policy. *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 255–261), Menlo Park, CA: AAAI Press.

Provost, F., & Buchanan, B. (1992b). Inductive strengthening: the effects of a simple heuristic for restricting hypothesis space search, in K.P. Jantke (Ed.), *Analogical and inductive inference (Proceedings of the 3rd International Workshop on Analogical and Inductive Inference)*, New York, NY: Springer-Verlag.

Provost, F.J., Buchanan, B.G., Clearwater, S.H., & Lee, Y. (1993). *Machine learning in the service of exploratory science and engineering: a case study of the RL induction program* (Technical Report ISL-93-6), Intelligent Systems Laboratory, University of Pittsburgh.

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

Quinlan, J. (1986b). Inductive knowledge acquisition: a case study, in J. Quinlan (Ed.), *Applications of expert systems* (pp. 157–173).

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

Quinlan, R. (1993). Presentation of new results on selecting from multiple learning algorithms at the Tenth International Conference on Machine Learning.

Rabinowitz, H., Flamholz, J., Wolin, E., & Euchner, J. (1991). NYNEX MAX: a telephone trouble screening expert. *Innovative Applications of Artificial Intelligence, 3* (pp. 213–230), Menlo Park, CA: AAAI Press.

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

Rendell, L., Seshu, R., & Tcheng, D. (1987). Layered concept-learning and dynamically-variable bias management. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (pp. 308–314), San Mateo, CA: Morgan Kaufmann.

Riddle, P. (1989). *Automating shifts of representation*, Ph.D. Thesis, Department of Computer Science, Rutgers University.

Ruff, R. (1990). *An empirical study into learning through experimentation*, Ph.D. Thesis, Department of Computer Science, Oregon State University.

Russell, S. (1986). Preliminary steps toward the automation of induction. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 477–484), San Mateo, CA: Morgan Kaufmann.

Russell, S., & Grosof, B. (1990). Declarative bias: an overview, in D.P. Benjamin (Ed.), *Change of representation and inductive bias* (pp. 267–308), Boston, MA: Kluwer Academic Publishers.

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

Samuel, A. (1963). Some studies in machine learning using the game of checkers, in E. Feigenbaum & J. Feldman (Eds.), *Computers and Thought* (pp. 71–105), New York, NY: McGraw-Hill.

Schaffer, C. (1993). Selecting a classification method by cross-validation, *Machine Learning*, 13(1):135–143.

Schlimmer, J. (1987). *Concept acquisition through representational adjustment*, Ph.D. Thesis, Department of

Information and Computer Science, University of California at Irvine.

Simon, H.A., & Kadane, J.B. (1975). Optimal problem-solving search: all-or-none solutions, *Artificial Intelligence*, 6:235–247.

Simon, H.A., & Lea, G. (1974). Problem solving and rule induction: a unified view, in L.W. Gregg (Ed.), *Knowledge and cognition* (pp. 105–128), Lawrence Erlbaum Associates.

Slagle, J.R. (1964). An efficient algorithm for finding certain minimum-cost procedures for making binary decisions, *Journal of the ACM*, 11(3):253–264.

Spears, W., & Gordon, D. (1991). Adaptive strategy selection for concept learning. *Proceedings of the First International Workshop on Multistrategy Learning* (pp. 231–246), Fairfax, VA: George Mason University.

Subramanian, D. (1989). *Toward a theory of justified reformulations*, Ph.D. Thesis, Stanford University.

Tcheng, D., Lambert, B., Lu, S., & Rendell, L. (1989). Building robust learning systems by combining induction and optimization. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 806–812), San Mateo, CA: Morgan Kaufmann.

Uhr, L., & Vossler, C. (1963). A pattern-recognition program that generates, evaluates, and adjusts its own operators, in E. Feigenbaum & J. Feldman (Eds.), *Computers and thought* (pp. 251–268), New York, NY: McGraw-Hill.

Utgoff, P. (1984). *Shift of bias for inductive concept learning*, Ph.D. Thesis, Rutgers University.

Utgoff, P. (1986). Shift of bias for inductive concept learning, in R. Michalski, J. Carbonell, & T. Mitchell (Eds.), *Machine learning, an artificial intelligence approach* (pp. 107–148), San Mateo, CA: Morgan Kaufmann.

Utgoff, P.E. (1989). Incremental induction of decision trees, *Machine Learning*, 4(2):161–186.

Weiss, S., & Indurkhya, N. (1991). Reduced complexity rule induction. *Proceedings of IJCAI-91* (pp. 678–684).

Wirth, J., & Catlett, J. (1988). Experiments on the costs and benefits of windowing in ID3. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 87–99), San Mateo, CA: Morgan Kaufmann.

Received November 15, 1993
Final Manuscript August 31, 1994