LAYERED CONCEPT - LEARNING AND DYNAMICALLY - VARIABLE BIAS MANAGEMENT

Larry Rendell, Raj Seshu, and David Tcheng

Department of Computer Science,
University of Illinois at Urbana-Champaign, Urbana, Illinois 61801

ABSTRACT

Concept learning is inherently complex. Without severe constraint or inductive "bias," the general problem is intractable. While most learning systems have been designed with built-in biases, these systems typically work well only in narrowly circumscribed problem domains. Here we present a model of concept formation that views learning as a simultaneous optimization problem at three different levels, with dynamically chosen biases guiding the search for satisfactory hypotheses. In this model, the partitioning of events into classes occurs through dynamic interactions among three layers: event space, hypothesis space, and bias space. This view of the induction process may help clarify the problem of learning and lead to more general and efficient induction systems. To test this model of meta-knowledge, a variable bias management system (VBMS) has been designed and partly implemented. The system will dynamically alter evolving hypotheses, concept representation languages, and concept formation algorithms by monitoring progress and selecting biases based on characteristics of the particular induction problems presented. VBMS is designed to learn the best biases for different types of induction problems. Thus it is robust (effective and efficient in many domains). The system can learn incrementally despite noisy data at any level.

I. INTRODUCTION

In theory the important problem of concept formation is simple: identify and describe useful classes of objects. Unfortunately, this inductive problem is inherently complex (Watanabe, 1969). In practice, all inducers, both human and mechanical. must be able to reduce the number of hypotheses by choosing the proper constraints or *biases* (Mitchell, 1980).

A. Mechanized Concept Formation

Class formation partitions a universe of objects, instances, or *events* into subsets.[1] A class is described by a *concept*. A candidate concept is a *hypothesis* (Mitchell, 1982). Given a domain, the universe of possible events is divided into those events that are consistent with the description of some underlying concept and those that are not. This *target concept* (e.g. a Boolean function over all events) may be discovered by searching "hypothesis space," or by building and modifying hypotheses in "event space" (Michalski, 1983).

1. To confound the problem, new objects or descriptions must sometimes be constructed, and it is the resulting universe that must be classified. This "problem of new terms" is discussed in (Michalski, 1983; Rendell, 1985).

While this "crisp" view of concept formation is prevalent in AI, a more refined view is useful in real-world environments. Since data are often uncertain, events may have degrees of class membership. Thus concept formation becomes the partitioning of the universe of events into graded *utility classes,* and instead of being a Boolean function, a concept becomes a multivalued, often probabilistic function (Rendell, 1986a; Zadeh, 1965). Moreover, since description languages are often deliberately constrained to control the difficulty, the concept may be imperfect; consequently hypotheses have degrees of credibility. So instead of being absolutely right or wrong, a hypothesis may also be assessed probabilistically (Rendell, 1986b; Watanabe, 1969).

B. Combinatorial Complexity

The practical problem faced in automated concept formation is managing the combinatorial explosion of hypotheses. The space of all hypotheses for a given problem contains every possible class description expressible in the language. Consider, for example, a 10×10 grid of bits which can be used to encode different symbols. Each bit on the grid can be on or off independently of the others, yielding a total of 2^{100} possible patterns or events. If we want to learn a symbol, there are $2^{(2^{100})}$ possible classifications or hypotheses to choose from. A naive "generate-and-test" inductive system would consider each one of the hypotheses; even algorithms such as candidate elimination (Mitchell, 1982) cannot tractably solve this problem.

We can reduce the combinatorial complexity by replacing the 100 primitive pixels by a reduced set of abstract features. For example, one feature might indicate the presence or absence of standard curves and strokes. If there were, for instance, only 6 features of 5 values each, there would be 5^6 or about 16,000 possible configurations. Yet even with this drastic simplification, the number of possible hypotheses is 2^{15625} — still well beyond the limits of practical computation.

To contain the combinatorial complexity, all learning systems employ one or more means of reducing the space of hypotheses. One simple method is to restrict the universe of events, as in our example, but there are several other ways. Techniques for pruning hypotheses provide inductive "bias."

C. Inductive Bias

Mitchell (1980) defined *bias* as "any basis for choosing one generalization over another, other than strict consistency with the observed training instances" (cf. Watanabe's (1969) "inductive ambiguity"). Bias encompasses all extra-evidential choices made to reduce the complexity of a learning problem.

These choices are often "hard coded", with the result that learning systems rarely perform well outside of narrowly circumscribed problem domains (such systems are "brittle"). This limitation can be overcome only by designing systems capable of dynamically altering their biases to accommodate new problems, as Utgoff (1986) has begun to do.

One distinction among biases is their *strength* (Mitchell, 1980; Utgoff and Mitchell, 1982; Utgoff, 1986). Strong biases cause an induction systems to exclude a relatively large proportion of possible hypotheses. Rendell (1986b) has quantified the notion of bias strength and used the measure to analyze some learning systems and the kinds of concepts they can manage; Haussler (1986) has formalized a relationship between bias strength and learnable concepts. In general, strong biases mean fast concept acquisition but they may miss the target concept; weak biases are more likely to include the concept but they usually retard learning to the point of infeasibility.

Although the notion of bias strength is a valuable one, our model of learning refines the problem along a different dimension: when and how to vary the bias. While Utgoff's (1986) variable-bias system dynamically invokes a weaker bias when a strong one fails, the decision is based on immediate experience. In contrast, we are concerned with accumulated experience and the connection between problems and the particular biases used to solve them (this is "meta-knowledge").

D. Scope of This Paper

This paper will examine the problem of dynamically variable-bias and the tools needed to address it. We shall detail the problem of bias from this perspective in the next section. Section III will describe our "three space" model of concept formation which facilitates a solution to the problem. Section IV will use this framework to develop the variable-bias management system, and Section V will present some preliminary results which support the model.

II. BIAS FLEXIBILITY AND BINDING TIMES

Concept learning systems have implemented inductive bias with varying degrees of flexibility and power. Depending on system design, bias may be decided by the user, or it may be (partially) determined by the program. When bias selection is mechanized, its degree of automation may vary with respect to the range of bias choices and the flexibility of their selection.

A. Fixed Bias

The most common approach is to use a *fixed bias*, which is "built-in" at system design time and cannot be altered without changing the program. One common fixed bias excludes many hypotheses by abstracting secondary objects to be the objects from which the induction system generalizes. In Samuel's (1963) checkers program, the primary or "primitive" objects are board configurations, but the abstract or learned objects are k-tuples of highly descriptive features (such as piece advantage) which compress the information. Another way to supply fixed bias is to restrict the language for hypothesis representation. For example, a logic-based language may confine hypotheses to those having few disjuncts (Michalski, 1983). By restricting the language of representation, systems limit expressible concepts; hence these systems speed processing but are applicable to some problems only.

B. Parameterized Bias

Some systems allow the user to instruct them to ignore or disfavor certain types of hypotheses. Because it can be altered at the beginning of a run, this is *parameterized bias*. In the AQ systems (Michalski, 1983), biases such as hypothesis simplicity are parameterized in the form of a "lexicographic evaluation functional" (LEF). An expression of hypothesis quality or preference, the LEF may be input by the user of AQ at run-time. For instance, a user might specify a preference for hypotheses having few disjuncts. The LEF thus biases the system towards "desirable" hypotheses while downgrading the less desirable. Lenat (1983) uses a slightly different approach to parameterized bias in Eurisko. In Eurisko a user can temporarily suspend the processing in order to fine-tune a system parameter.

C. Dynamic Bias

A still more flexible induction system may be capable of altering its biases during the course of execution. This form of bias is *(dynamically) variable*. Unlike a parameterized bias, a variable bias does not require the user to make decisions; rather the system will "set" its own bias according to its experience. One step toward mechanizing the selection of a bias is Utgoff's system called "search for a better bias," or STABB (Utgoff, 1986). While solving a given problem, STABB is capable of forming a new disjunct to add to the hypothesis representation language. This expansion takes place if hypotheses using fewer disjuncts have been rejected by evidential criteria.

D. More Powerful Variable Bias Management

Despite its flexibility, STABB is limited to altering one kind of bias. Insofar as the control strategy involves a fixed ordering of choices, we might say the hypotheses are ordered by design-time criteria. In contrast, more powerful variations on the variable-bias scheme may not rely on a fixed ordering strategy. Instead, they would be capable of learning through experience just when different biases are appropriate. Not only would dynamic-bias management choose biases according to problem type, these schemes could have a greater range of choices. By using accumulated knowledge, they would systematically alter their biases, which would include not only aspects of representation such as features for event description, but also aspects of the inductive algorithm itself (e.g., hypothesis transformation operators might be selected to compose an algorithm dynamically). The result would be an inductive system that is adaptive, efficient, and *robust* (i.e. effective over a wide variety of problem domains).

This is the motivation for the *variable-bias management system* (VBMS). Biases are dynamically located and adjusted according to problem characteristics and past experience with similar problems. Through exposure to different types of problems, VBMS induces problem classes and identifies techniques (biases) appropriate for each class. With experience, VBMS will evolve into a general learning system capable of identifying and effectively learning diverse classes of problems. The fundamental idea behind VBMS is the use of multiple layers of learning (see Buchanan et al., 1978).

III. A THREE-SPACE MODEL FOR LEARNING

In this section, we shall view concept learning as a parallel inductive search through multiple spaces. The model will be used to explore interrelationships between spaces, to clarify the problem of concept learning, and to explain and develop VBMS (Section IV).

The layers in our model are associated with three distinct spaces: event space, hypothesis space, and bias space. Event space orders the events to which the learning system might be exposed. Hypothesis space describes all possible partitions of the events into concept classes. Bias space specifies the potential combinations of constraints that may be imposed upon a hypothesis space.

A. Event Space

Inductive systems learn by extracting information from a set of *events*, which are the "ground objects" or basic cases to be formed into classes and described by the target concept. Events are provided to a learning system as descriptions, usually consisting of a list of features (such as shapes of figures or strategic board positions). The set of all features over which a system operates constitutes a feature space in which each feature forms a distinct dimension. Each point in this space represents a possible event. Since this space defines all possible events which the system might ever encounter, the space is called *event space* or *E-space*.

We can consider similarity of events within E-space as a relevant criterion for assessing the ultimate "utility" of events. By *utility*, we mean the degree of concept membership. We can view a concept as a surface over E-space, where the shape of the surface is defined by the utility function. If a point in event space has a high utility value, we may expect neighbors of that point also to be of relatively high utility.

E-space proximity underlies many efficient induction methods such as hyperrectangle creation, discriminant analysis, conceptual clustering, etc. (Rendell, 1986b). Proximity in event space is useful, however, only if the utility surface is regular. A *regular* space can be visualized as a smooth surface without abrupt deformations. Operationally, a regular utility surface means that generalization and specialization operators may proceed more efficiently (Rendell).

B. Hypothesis Space

The purpose of a concept learning system is to form a description extending the set of positive events to predict others. But for any given set of events, there are many potential extensions. If the utility is simply binary, and if E-space has only 1000 possible events, 100 of which are known to be positive, then there are $2^{(1000-100)} \simeq 10^{271}$ valid extensions.

A given system will attempt to select and describe the "correct" extension, where correctness depends upon the goals of the system. Each description is a *hypothesis* for what the correct one might be. Some hypotheses, however, are more "credible" than others, i.e. they more closely approximate the correct (target) concept. Thus, each hypothesis can be assigned a graded value called its *credibility*, which is analogous to the utility of an event in E-space. In our example of 1000 events, the induction problem is to guess the binary utility value of the 900 unknown events. The greater the number of correct guesses, the better the (evidential) credibility.[2]

Each hypothesis may be viewed as a function u. The domain of u is the set of points in E-space; the range of u is a real number between 0 and 1 reflecting the degree of concept membership — u is the utility function. Each hypothesis is a complete surface over E-space which maps every point in E-space to a specific utility. A hypothesis can therefore be viewed as a *surface over E-space*. The credibility of a hypothesis depends upon how closely the hypothetical utility surface matches the correct utility surface of the concept.

Just as we may think of the set of events as constituting E-space, we may also think of the set of hypotheses as being organized into its own space. Given a particular representation for concepts and hypotheses, the space covering all possible hypotheses is called *hypothesis space* or *H-space*.

Inductive operators which generalize, specialize, or otherwise transform hypotheses allow systems to move from one H-space point (hypothesis) to another. For example, in an H-space based on logic, applying an operator that replaces a constant with a variable results in movement to a more general hypothesis.

The size and structure of H-space depends on the event and hypothesis description languages. Consider the problem of symbol recognition. If the experimenter uses 100 primative features (e.g. pixels), then an impractically large $2^{[2^{100}]}$ yet highly expressive H-space results. To reduce the enormous H-space resulting from a detailed event language, the experimenter could, e.g., restrict the hypothesis language to some logic function over a limited number of pixels. If instead, the experimenter used abstract features (e.g. x_1 = number of lines, x_2 = number of curves), then a much smaller H-space would result. Even so, feasible search requires techniques such as hill climbing, which take advantage of regularity or smoothness in the function being optimized.

Regularity in H-space can be defined like regularity in E-space, except that we use a credibility surface instead of utility. In a regular H-space, we may say something about the credibility of a hypothesis' neighbors once we know something about the hypothesis' credibility. Regular credibility surfaces in H-space mean that we can move to neighboring hypotheses without experiencing significant variations in credibility. If, on the other hand, there is no discernible regularity within the H-space, then a new bias may be employed to reevaluate hypothesis credibility.

C. Bias Space

The process of concept learning is equivalent to a search through hypothesis space — the goal is to pick the "correct" hypothesis to classify the set of events. The choice of what constitutes the correct hypothesis depends on several factors. Evidential or E-space information will heavily influence the decision. Just as important, though, are the extra-evidential factors that constrain the potential form of the hypotheses. These factors are the system's *biases*. Different systems have different sets of biases and thus have different hypothesis spaces in which they operate.

---

2. Since the time required to discover evidential credibility is usually extreme, a learning system is usually designed to estimate an additional component that is faster to compute — the "extra-evidential" credibility. Since refined comparisons are helpful, the credibility should be graded rather than all-or-none (for other reasons see Rendell, 1986b).

In the symbol recognition problem, many possible biases could constrain the event or hypothesis description language. To describe grids, suppose we have 3 available representations: (1) binary pixel values, (2) shape frequencies (e.g. number of lines, etc.), and (3) line angle distributions. Suppose also that we have 3 inductive algorithms available which variously represent hypotheses as (1) linear discriminant functions, (2) prototypes, or (3) utility regions. Assuming the event descriptions are compatible with the inductive algorithms, our possible choices of event and hypothesis representation languages define a simple 3×3 "bias space."

In the same way that we proposed an H-space over all hypotheses, we can envision a third space over all biases. The collection of all possible biases make up this new space, called *bias space* or *B-space*. Each bias exists as a point in B-space.[3] Just as we viewed a hypothesis as both a point in H-space and a utility surface over E-space, we may view a particular bias as both a point in B-space and as a credibility surface over H-space. A credibility surface over H-space is a function that takes a hypothesis from H-space and maps it onto a credibility value. In other words, each hypothesis defines a utility surface over E-space, and each bias defines a credibility surface over H-space. Operationally, the credibility surface over H-space actually represents an ordering of all potential hypotheses to be evaluated in H-space. This credibility surface is also (by definition) a point in bias space.

Induction systems that are capable of dynamically altering biases work with a further measure. Just as a hypothesis has a credibility, a bias has a *belief* associated with it. Belief is the induction system's estimate of the "goodness" of a bias, just as the credibility is an estimate of the "goodness" of a hypothesis, and the utility is an estimate of the "goodness" of an event.

D. Relationships Between Spaces

Every point in E-space represents an event that the system might possibly encounter; every point in hypothesis space is a utility function over the events in E-space; every point in bias space is a credibility function over hypothesis space. Choosing a particular point in hypothesis space is tantamount to characterizing the utility of *every* point in event space. Choosing a particular point in bias space is tantamount to characterizing the credibility of *every* point in hypothesis space.

Many induction systems use the credibility surface over hypothesis space to guide the search for hypotheses (Rendell, 1987). Similarly, induction systems with variable biases should be able to use a belief surface over bias space to guide the search for new algorithms and/or representation languages. Thus, in a manner analogous to identifying correct hypotheses through a search of hypothesis space, the "correct" bias for a given problem domain may be found by searching bias space.

To say that bias space is regular means that the average difference in belief between neighboring points is small. In that case, a proper distance measure can allow a variable-bias system to perform what is essentially hill-climbing through bias space. Unfortunately, two biases which may be very close together in the context of one problem may not yield sufficiently similar results in other domains. Even small variations in problem characteristics may produce cases in which two biases perform similarly in one situation but differently in another. However, if the experimenter (or the induction system) knows or infers that two problems are sufficiently similar, then hill-climbing in bias space can be valuable. In that case, knowledge of the proximity of different biases in the context of one problem can provide the basis for hill-climbing in the second problem.

In all three spaces, the same essential phenomenon permits the same basic techniques. Regularity or smoothness in a certain function allows efficient methods such as hill-climbing. Depending on the level of learning, the function may be called *utility, credibility,* or *belief,* but the important general phenomenon of proximate similarity is responsible for efficiency at all three levels. These ideas are expanded and analyzed in (Rendell, 1986b, 1987); the learning system using them is developed below.

IV. VARIABLE-BIAS MANAGEMENT

In Section II we saw that to avoid brittleness and extend efficacy, we need more flexible learning. According to the three-space model, flexible learning can be viewed as a parallel search across event, hypothesis, and bias spaces. The variable-bias management system VBMS is a realization of this concise multiple-space model. By controlling movement between the three spaces, the VBMS is designed to learn the most effective techniques of induction for a wide range of problem classes.

Ideally, a learning system should be able to select its own biases. Biases (including representations, algorithms, and components of each) should depend on problem characteristics. Although this ability can be "hard-coded" into the system by the experimenter, doing so yields a system that performs well for a few problems familiar to the researcher but with no ability to learn from mistakes or adapt to new classes of problems. A more flexible approach is to let the system learn the concept of appropriate bias selection from scratch — as a direct result of problem solving experience. The VBMS begins with no knowledge of the appropriateness of biases, but gradually induces problem classes along with the corresponding biases most useful for the effective learning of the problems in each class. To explain the operation of the VBMS we begin with a discussion of a simplified, naive approach.

A. Naive Bias Management

One naive approach is to try all available biases for a given problem, calculate the average effect of each bias (over all problems encountered), and then use this general knowledge bias space to select future biases. Such a naive approach might consist of the following components: (i) a set of bias points (to determine a space of biases); (ii) a general belief table (to assign a belief β to each bias point); and (iii) a credibility measure (to judge the correctness of hypotheses).

A point in bias space is a choice of inductive algorithm, representation language, and any relevant parameters to the algorithm or language (e.g. number of disjuncts, splitting criterion, etc.). The system records its belief β in each bias in a general belief table (GBT) which is simply a list of bias points

3. "Bias" will be used to refer to either a single bias or a set of biases. Thus, while a system may assume several biases, the set of these biases will be called the system's bias.

and corresponding β's. For example a simple bias space might have two-dimensions: (1) representation (e.g. number of disjuncts), and (2) algorithm (e.g. generalization or specialization). The general belief table might gradually assign greater beliefs to few disjuncts, perhaps because learning is faster and just as accurate.

One natural measure of β is the credibility of the best hypothesis produced by a particular bias choice, relative to the credibility of hypotheses produced by other biases. Many credibility measures exist (both evidential and extra-evidential), and the only restriction in choosing a credibility metric is that it be uniformly applicable to any hypothesis generated by the system. For instance, if the problem is character recognition, an appropriate credibility measure would be how quickly and accurately the hypothesis classifies characters.

A bias management system operates in two modes: learning and performance. In the learning mode, the system tests all available biases on each given problem. Testing a bias entails running the associated inductive algorithm, measuring the credibility of hypotheses generated, and retaining the most credible ones. After all biases have been tried, the system scales the credibilities of retained hypotheses to fit in the interval [0,1], with the better hypotheses earning values closer to 1. These normalized credibilities represent the system's belief β in each bias point, relative to a specific problem.

As outlined in the three-space model (Section III), these bias points and β's can be visualized as forming a surface over bias space with peaks representing relatively good biases. Since initial implementation of a bias management system would involve only a limited number of bias points (compared to the space of all possible biases), this "surface" is more conveniently represented as a belief table. A problem belief table (PBT) contains all biases explored for a specific problem and their estimated β's. Similarly the average of all PBT's created in the system's lifetime forms a general belief table (GBT).

In the performance mode, the naive system simply tries the biases in its GBT in descending order of their estimated belief until an acceptable hypothesis is found or the rate of credibility improvement per bias tried falls below some threshold.

This bias management is more flexible than Utgoff's STABB, which has a limited set of biases and does not adjust their order. On the other hand, our naive algorithm is still limited. Its major fault is that it tries biases according to their *average* performance over all problems experienced. This approach makes no use of problem characteristics when selecting biases. Its behavior is analogous to a doctor who always prescribes aspirin regardless of the symptoms because he has found the drug to be effective in most situations. Even if this system were to encounter a new problem identical to a previously solved problem it would still use its general belief table rather than the more appropriate problem belief table.

B. Improved Bias Management

While naive bias management might be useful for certain applications, it would be too coarse. A cure is to perform induction on bias space itself, and allow the system to learn how to apply different biases to different types of problems. This meta-level learning is the basis of the variable-bias management system VBMS.

Intuitively, any similarity between a given problem and previously solved problems should influence bias selection. If the similarity is high, we are tempted to give some weight to the PBT (problem belief table) generated from related past problems when selecting biases for the new problem. If the similarity is low, the system may be justified only in using its general knowledge of bias space found in its GBT. The use of similarity to select biases assumes that similar problems will have similar solutions. Thus the appropriate choice of a similarity measure is crucial to the operation of a more flexible bias management system. The VBMS uses a *dynamic similarity measure that evolves with experience.* This dynamic assessment of similarity is a novel feature of the VBMS and should result in great flexibility.

To associate problem characteristics with effective biases, we need to introduce the idea of a problem space. Problem space is similar to event space except that its dimensions are global features of the problem rather than descriptions of events. Problem characteristics are user-defined and should be applicable to all problems presented to the system. For instance, if the VBMS consists of algorithms that process feature vectors, then potentially useful problem characteristics include the number of training events, the reliability of training events, the number of features per event, the grain size of features, and other properties of features.

Over time, the system partitions the points in problem space into regions (problem classes) whose problem points have similar solutions (PBT's). In other words, problem points in the same region have similar bias beliefs or PBT's. The formation of these regions involves a region belief table (RBT) to store regional beliefs. Like the GBT of the naive system, an RBT is an average of all PBT's belonging to problems within the region. For example, regions of problem space where the grain size is large might have a strong belief in few disjuncts. Problems within the same region are considered "similar" with respect to their solutions and bias beliefs. (Similarities in values, tables, or functions can be used to form and modify regions; see Rendell, 1985.)

Every new problem attempted by the VBMS is associated with a point in problem space. When selecting biases, the system finds the region containing the new point in problem space, and uses the region belief table to select initial biases. Biases in the RBT are tried in order of decreasing belief until an acceptable hypothesis is found or the rate of credibility improvement falls below some threshold. If this approach fails, the GBT is then used to select a bias.

At first VBMS exhaustively searches bias space for each new problem and learns only general knowledge of the belief surface (i.e. a GBT). As more problems are attempted, VBMS gradually learns relationships between problem characteristics and effective biases. This knowledge resides in the problem space regions and their associated RBT's.

The reliance of VBMS on multiple iterations of concept formation suggests inefficiency. The system might be too slow if it had to construct a new region belief table for each new learning problem. But since this meta-level knowledge (RBT) is accumulated for all problems encountered, the procedure is reasonable (as Section V begins to show). A more complete description of the VBMS algorithm appears in (Rendell et al., 1987b).

V. IMPLEMENTATION AND EXPERIMENT

We have begun to implement and test the VBMS. To explain details, let us first consider some concrete forms of bias.

A. Kinds of Bias

From the standpoint of computer implementation there are two basic manifestations of bias: representational (for description of events and concepts), and algorithmic (for construction, transformation, and verification of hypotheses).

1. Representational Bias

Given a language such as DNF logic, the specific elements of the language (e.g. function symbols or attributes) and constraints on it (e.g. few disjuncts) are the biases that the user or system may control. The manifestation of bias as a variable number of disjuncts (Utgoff, 1986) is a special case of an imposed model: In uncertain environments, "number of disjuncts" becomes the number of peaks in the utility function. Additional constraints may confine the functional form (e.g. a linear combination of features — see Rendell, 1983). While functional forms such as the number of disjuncts are straightforward to modify, not all representational biases are easy to implement. For example, the choice of features is a complex research problem (Porter, 1986).

2. Algorithmic Bias

Given a hypothesis language having a particular representational bias, a concept learning algorithm is designed to search in an associated space for credible hypotheses that approximate the target concept. Since the learned concept may depend on details of the search, algorithms and algorithm components are also biases. Just as hypotheses space contains the desired concept, "algorithm space" contains the desired algorithm for finding it. (We could think of algorithm space as being the subpace of the entire bias space that has to do with algorithms only.) Just as the target concept for the current domain problem is extracted from hypotheses space, a well-behaved algorithm for the current learning problem is extracted from algorithm space.

Depending on our representation of algorithm space, it could be more or less grainy. A very grainy algorithm space might contain a few fixed concept learning systems, such as AQ, ID3, PLS1, etc. (Michalski, 1983; Quinlan, 1983; Rendell, 1983). In contrast, a refined algorithm space might contain system *components* (such as operators for hypothesis transformation). In the simple (grainy) case, a completed algorithm would be selected as a unit; in the complex (refined) case, the algorithm would be constructed from its components.

Table I. Run Times for Three Induction Algorithms.*

# Attributes	#Events	Aq15 Time (μ1)	Assistant Time (μ2)	PLS1 Time (μ3)
8	150	41.7 (.16)	6.7 (1.00)	31.3 (0.21)
10	150	115.1 (.11)	12.8 (1.00)	36.0 (0.36)
12	150	358.2 (.06)	20.7 (1.00)	41.2 (0.50)
14	150	1013.8 (.04)	49.0 (0.89)	43.7 (1.00)
16	150	1340.4 (.04)	65.9 (0.76)	49.8 (1.00)
18	150	1759.5 (.03)	83.5 (0.71)	59.4 (1.00)
18	140	1595.5 (.03)	81.8 (0.58)	47.8 (1.00)
18	130	1694.6 (.03)	78.2 (0.60)	46.9 (1.00)
18	120	1536.7 (.03)	73.2 (0.61)	45.0 (1.00)
18	110	1395.0 (.03)	72.8 (0.50)	36.3 (1.00)
18	100	1172.4 (.03)	66.3 (0.51)	34.1 (1.00)

* "Time" is user time in seconds. The (normalized) μ is the ratio of the given algorithm's performance to the performance of the fastest algorithm.

B. Initial Implementation

To initiate testing of variable-bias management, we wanted to begin with one of the simpler kinds of bias. We decided to begin with algorithmic biases and have the VMBS try to choose an entire learning system from a coarse algorithm space. Unlike Utgoff's (1986) STABB, VMBS bases its choices on characteristics of the problem domain (although the characteristics so far are simple and syntactic).

1. First Experiment: Bias as Algorithm

In the first experiment VMBS selects one of three learning systems AQ15, ASSISTANT, and PLS1 (which are dissussed in Rendell et al., 1987a). The choice is based on the behavior of these three systems as a function of number of training events and number of features. In other words, problem space here is only two-dimensional: the number of events is on one axis, and the number of features in each event is on the other.

Table I shows results involving a lymphography data base, using the learning systems AQ15, ASSISTANT, and PLS1. Initially, VBMS tries each algorithm on each problem, until sufficient experience is accumulated (at this time the criterion for "sufficient experience" is user-supplied, although full automation will simply be information-theoretic).

The credibility measure should reflect performance, and should be include concept accuracy and processing time (measures for these are standard — see Rendell et al., 1987a). To keep the first experiment simple, our uniform measure of credibility is just the number of user-CPU-seconds (on a VAX780 running under UNIX — all programs were written in Pascal). In this case we normalize credibility μ by just taking the quotient: number of seconds used by the fastest algorithm over the number of seconds used by the given algorithm. In this case, with only three algorithms, a triplet of μ-values is associated with each point in problem space. Call these values μ_1 (AQ15), μ_2 (ASSISTANT), and μ_3 (PLS1), and the resulting μ-vector $\boldsymbol{\mu}$.

VBMS divides problem space by splitting it into orthogonal rectangles, making the splits having the highest dissimilarity rating (like the basic PLS1 algorithm). To calculate the dissimilarity rating, VBMS first averages the μ's for each of the three algorithms in the proposed regions. Call these averages for two tentative subrectangles $\hat{\mu}_i$ and $\hat{\mu}'_i$ ($1 \leq i \leq 3$). Next, VBMS calculates

$$d = \log \left[\sum_i \left(\max \{ \hat{\mu}_i / \hat{\mu}'_i, \hat{\mu}'_i / \hat{\mu}_i \} \right) \right].$$

(This is based on an information-theoretic measure and should generally include an error term; see Rendell, 1983.)

Number of Attributes	150 Events	140 Events	130 Events	120 Events	110 Events	100 Events
18	(.03, .71, 1)	(.03, .58, 1)	(.03, .60, 1)	(.03, .61, 1)	(.03, .50, 1)	(.03, .51, 1)
16	(.04, .76, 1)					
14	(.04, .89, 1)					
12	(.06, 1, .50)					
10	(.11, 1, .36)					
8	(.16, 1, .21)					

Figure 1. Choosing biases dynamically according to problem characteristics. Each element of a vector (here a triple) represents the performance of a different bias. In this simple case, performance depends on the number of training events and the number of attributes (giving a two-dimensional problem space). VBMS splits regions in problem space if different biases markedly affect algorithm performance. Here VBMS forms three distinctive regions.

2. Results of Initial Experiment

As shown in Fig. 1, VBMS split problem space into 3 rectangles. The first split is perpendicular to the number-of-attributes axis. The eight points with 14 or more event attributes have an average μ of (0.03, 0.65, 1.00), and the 3 other points have an average μ' of (0.11, 1.00, 0.36). The dissimilarity value is $d = \log 8 \simeq 3$.

The second split occurs with problems of 150 events (and 14 or more attributes) and problems with fewer events (but 14 or more attributes). These two regions have average μ's of (0.03, 0.56, 1.00) and (0.04, 0.79, 1.00). The dissimilarity value is $\log 2.6 \simeq 1.6$. If we insist that each region have at least two points, none of the other possible splits have dissimilarity ratings higher than $\log 1.2 < 0.5$.

Henceforth, VBMS expects PLS1 to be twice as fast as ASSISTANT in problems with fewer than 140 events and more than 12 attributes, and it expects PLS1 to be only about 20% faster for problems with 150 events and more than 12 attributes. Finally, in problems with fewer than 14 attributes, VBMS expects ASSISTANT to run three times as fast as PLS1.

In future problems, VBMS can use the best algorithm for the problem to perform faster learning (the differences can be much larger than 3 to 1 — see Rendell et al., 1987a). The overhead for running VBMS is insignificant. Since VBMS uses fewer resources, it performs better than its subsidiary algorithms. VBMS dynamically choses biases and learns to associate them with classes of problems.

VI. SUMMARY AND CONCLUSIONS

Since induction is so complex, concept formation is feasible only by reducing the search space through the use of selective biases. Biases appear in many different forms (e.g. abstracted features, acceptable concept descriptions, etc.) and are usually fixed into the design of a system. Since any fixed bias is too restrictive for some problems and too slow for others, biases should be dynamic — for generally efficient and effective (robust) learning, we need better methods for managing bias mechanically.

Previous work on dynamic bias has been quite limited. Utgoff's (1986) STABB includes a variable bias, but only in a form that does not learn to associate bias with knowledge about problems.

We have outlined a robust multiple-space model of learning and proposed a design for a variable-bias management system VBMS. In this model of knowledge and meta-knowledge, learning can be viewed as a parallel search across event, hypothesis, and bias spaces. The VBMS is specifically designed to learn biases and to induce their relationships to classes of problems, and thereby to support flexible learning across different problem domains. Unlike most other learning systems, the VBMS learns at different levels.

An inductive algorithm performs better when it outputs hypotheses of higher credibility using fewer resources. Even in initial implementation involving a coarse "algorithm space," we have shown that VBMS performs better any of its subsidiary algorithms alone. By dynamically adjusting its bias, VBMS can select and use a strong bias appropriate to a given induction problem. Thus it can combine the efficiency of a strong bias with the generality of a weak one.

VBMS is a robust framework for probabilistic, multilayered learning. Extensions include elaboration of problem space and bias space using analogy and other semantic information. These and other refinements (and algorithms for the system) are discussed in (Rendell et al., 1987b).

REFERENCES

Buchanan, B.G., Johnson, C.R., Mitchell, T.M., and Smith, R.G., "Models of Learning Systems." In Belzer, J. (Ed.), *Encyclopedia of Computer Science and Technology*. 11 (1978), pp. 24–51.

Haussler, D., "Quantifying the Inductive Bias in Concept Learning." In *Proc. Fifth National Conference on Artificial Intelligence*. Philadelphia, PA, August, 1986, pp. 485–489.

Lenat, D.B., "The Role of Heuristics in Learing by Discovery: Three Case Studies." In Michalski, R.S. et al (Eds.), *Machine Learning: An Artificial Intelligence Approach*. Tioga, 1983, pp. 243–306.

Michalski, R.S., "A Theory and Methodology of Inductive Learning." In Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (Ed.), *Machine Learning: An Artificial Intelligence Approach*. Tioga, 1983, pp. 83–134.

Mitchell, T.M., "The Need for Biases in Learning Generalizations." Technical Report CBM-TR-117, Dept of Computer Science, Rutgers University, May 1980.

Mitchell, T.M., "Generalization as Search." *Artificial Intelligence*. 21 (1982), 203–226.

Porter, B., "PROTOS: An Experiment in Knowledge Acquisition for Heuristic Classification Tasks." *In Proc. International Meeting on Advances in Learning*, Les Arcs, France, August, 1986, pp. 159–174.

Quinlan, J.R., "Learning Efficient Classification Procedures and their Application to Chess End Games." In Michalski, R.S. et al (Ed.), *Machine Learning: An Artificial Intelligence Approach*, Tioga, 1983. pp. 463–482.

Rendell, L.A., "A New Basis for State-space Learning Systems and a Successful Implementation." *Artificial Intelligence*. 20:4 (1983) 369–392.

Rendell, L.A., "Substantial Constructive Induction Using Layered Information Compression: Tractable Feature Formation in Search." *In Proc. Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August, 1985, pp. 650–658.

Rendell, L.A., "Induction, Of and By Probability." In Kanal, L.N. & Lemmer, J (Eds.), *Uncertainty in Artificial Intelligence, 1986a*, pp. 429–443.

Rendell, L., "A General Framework for Induction and a Study of Selective Induction." *Machine Learning*. 1:2. (1986b), 177–226.

Rendell, L., "Layered Concept Learning and Its Advantages." University of Illinois Report UIUCDCS-R-87-1324, 1987.

Rendell, L.A., Benedict, P., and Cho, H., "Concept Acquisition from examples: Measurement of System Performance and Suggestions for Improved Design." University of Illinois Report UIUCDCS-R-87-1315, 1987a.

Rendell, L.A., Seshu, R., and Tcheng, D., "More Robust Concept Learning Using Dynamically-Variable Bias." In *Proc. Fourth International Workshop on Machine Learning*, 1987b.

Samuel, A.L., "Some Studies in Machine Learning using the Game of Checkers." In Feigenbaum, E.A. and Feldman, J. (Ed.), *Computers and Thought*, McGraw-Hill, 1963, pp 71–105.

Utgoff, P.E., "Shift of Bias for Inductive Concept Learning." In Michalski, R.S. et al (Ed.), *Machine Learning: An Artificial Intelligence Approach Vol. II*. Tioga, 1986. pp. 107–148.

Utgoff, P.E., and Mitchell, T.M., "Acquisition of Appropriate Bias for Inductive Concept Learning." In *Proc. National Conference on Artificial Intelligence*. Pittsburgh, PA, August, 1982, pp. 414–417.

Watanabe, S., *Knowing and Guessing: A Formal and Quantitative Study*. Wiley, 1969.

Zadeh, L.A., "Fuzzy Sets." *Information and Control*. 8 (1965), 338–353.