Quantifying Inductive Bias: AI Learning Algorithms and Valiant's Learning Framework

David Haussler*
Department of Computer Science, University of California, Santa Cruz, CA 95064, U.S.A.

Recommended by Tom Mitchell

ABSTRACT

We show that the notion of inductive bias in concept learning can be quantified in a way that directly relates to learning performance in the framework recently introduced by Valiant. Our measure of bias is based on the growth function introduced by Vapnik and Chervonenkis, and on the Vapnik-Chervonenkis dimension. We measure some common language biases, including restriction to conjunctive concepts, conjunctive concepts with internal disjunction, k-DNF and k-CNF concepts. We also measure certain types of bias that result from a preference for simpler hypotheses. Using these bias measurements we analyze the performance of the classical learning algorithm for conjunctive concepts from the perspective of Valiant's learning framework. We then augment this algorithm with a hypothesis simplification routine that uses a greedy heuristic and show how this improves learning performance on simpler target concepts. Improved learning algorithms are also developed for conjunctive concepts with internal disjunction, k-DNF and k-CNF concepts. We show that all our algorithms are within a logarithmic factor of optimal in terms of the number of examples they require to achieve a given level of learning performance in the Valiant framework. Our results hold for arbitrary attribute-based instance spaces defined by either tree-structured or linear attributes.

Introduction

The most extensively investigated learning task in artificial intelligence is that of learning a single concept from examples. For example, one might consider the task of learning to distinguish edible mushrooms from non-edible mushrooms by looking at preclassified examples of actual mushrooms (see e.g. [35]). In this task we select a set of mushroom attributes (e.g. color, shape and size) and attempt to find a rule that distinguishes between edible and non-edible mushrooms expressed in terms of these attributes (e.g. edible$\Leftrightarrow$((*color* = *red or orange*) *and* (*size* = *small*)) *or* . . .). The set of attributes selected determines

*The author gratefully acknowledges the support of ONR grant N00014-86-K-0454.

the *instance space* used by the learning algorithm, and the type of expression allowed in specifying the rule determines the *hypothesis space* used by the algorithm.

In any realistic learning application, the entire instance space will be so large that any learning algorithm can expect to see only a small fraction of it during training. From this small fraction, a hypothesis must be formed that classifies all the unseen instances. If the learning algorithm performs well then most of these unseen instances should be classified correctly. However, if no restrictions are placed on the hypothesis space and no "preference criterion" [24] is supplied for comparing competing hypotheses, then all possible classifications of the unseen instances are equally possible and no inductive method can do better on average than random guessing [26]. Hence all learning algorithms employ some mechanism whereby the space of hypotheses is restricted or whereby some hypotheses are preferred a priori over others. This is known as *inductive bias*.

The most prevalent form of inductive bias is the restriction of the hypothesis space to only concepts that can be expressed in some limited concept description language, e.g. concepts described by logical expressions involving only conjunction (see e.g. [6, 10]). A still stronger bias can be obtained by also introducing an a priori preference ordering on hypotheses, e.g. by preferring hypotheses that have shorter descriptions in the given description language (see e.g. [24, 33]). While many forms of bias have been used, up to this point there has been no generally agreed upon language-independent *measure* of the strength of a bias, in particular, a measure that relates the strength of a bias to the performance of learning algorithms that use it, so that it will be useful in analyzing and comparing learning algorithms. This paper proposes such a measure, and demonstrates how it can be used to compare and prove performance results for learning algorithms.

We measure bias with a combinatorial parameter defined on classes of concepts known as the *growth function* [43]. A theory and methodology of pattern recognition based on this function has been developed by Vapnik [42]. Applications of the theory to linear separators and Boolean circuits, and its relation to the preference for simpler hypotheses are discussed in [30]. The present work can be viewed as an extension of this methodology to concept learning problems in artificial intelligence.

The growth function of a hypothesis space can be used to define its *Vapnik–Chervónenkis dimension*, a combinatorial parameter closely related to the notion of *capacity* introduced in [9]. Extending the results of [42], in [5, 12] it is shown that this parameter is strongly correlated with learning performance as defined in the learning framework introduced by Valiant [21, 39–41]. We adopt this framework here as well.

The salient feature of the Valiant framework is that it only requires that the learning algorithm produce a hypothesis that with high probability is a good approximation to the target concept. It does not demand that the learning algorithm identify the target concept exactly. Angluin has called this framework "probably approximately correct" identification [1]. By adopting this weaker performance criterion, we are able to show that a number of simple learning algorithms actually perform near optimally in terms of the number of training examples they need. These algorithms include the "classical" algorithm for conjunctive concepts, and variants of this algorithm for related target classes.

In the Valiant framework, a training sample is created by drawing instances from the instance space independently at random according to some fixed probability distribution, and labeling them "+" or "−" according to whether or not they are instances of the target concept. Each such labeled instance is called a (*random*) *example* of the target concept. The *error* of a hypothesis is defined as the probability that it will disagree with a random example of the target concept drawn according to the same fixed probability distribution used to generate the training sample. Thus, if we are trained to recognize edible mushrooms on the west coast of the United States, we expect the rule we learn to work well in west coast forests, but not necessarily in east coast forests.

A good approximation to the target concept is a hypothesis with small error. Thus formally, the Valiant criterion demands that a learning algorithm produce a hypothesis that with high probability has small error with respect to a given probability distribution and target concept. A class of target concepts is considered learnable by the algorithm only if this happens for any target concept in the class and *any probability distribution on the instance space*. Thus, while the framework is probabilistic, it is not tied to any particular probability distribution or even to any type of distribution, and hence it provides an extremely robust performance guarantee.

Two measures of learning complexity are relevant in this framework. The first is *sample complexity*. This is the number of random examples needed to produce a hypothesis that with high probability has small error. As above, the sample complexity of a learning algorithm on a given target class is defined by taking the number of random examples needed in the worst case over all target concepts in the class and all probability distributions on the instance space. For each of the learning algorithms we present, we show that the sample complexity is within a poly-logarithmic factor of optimal.

The second performance measure is *computational complexity*, which we take as the worst-case computation time required to produce a hypothesis from a sample of a given size. We show that each of the learning algorithms we present has computational complexity polynomial in the sample size and in the number of attributes that define the instance space.

The paper is organized as follows. In Section 1 we define instance spaces on tree-structured and linear attributes, and we define various hypothesis spaces on such instance spaces. In Section 2 we take a new look at Mitchell's *version* *space* framework for learning concepts from examples [28], here from a probabilistic point of view. Mitchell defines the version space as the set of all hypothesis in the hypothesis space that are consistent with a given set of examples. We show (Lemma 2.2) that by using a hypothesis space that is strongly biased and by drawing independent random examples, the version space will shrink very rapidly, with high probability, to a set of hypotheses that cluster around the target concept in the sense that their errors are small relative to the target concept. For this initial result, bias is measured in terms of the size of the hypothesis space (see also [28, 32]). The result is then refined in Section 3 (Theorem 3.3) when we introduce the growth function as a measure of bias.

In Sections 4 to 7 we use these results to analyze the performance of several learning algorithms. We first consider what we call the classical algorithm for learning conjunctive concepts (Algorithm 4.1). This algorithm produces the unique maximally specific conjunctive hypothesis consistent with the training sample. Corollaries 4.5 and 4.8 provide bounds on the learning performance of this algorithm. The latter results show that its sample complexity is within a logarithmic factor of optimal (see also [12]).

In Section 5 we consider the problem of learning simple (i.e. syntactically short) conjunctive concepts on instance spaces with many attributes. We adapt the greedy heuristic for set cover [18] to simplify the hypothesis produced by the classical algorithm. The result is a learning algorithm (Algorithm 5.2) that has sample complexity within a poly-logarithmic factor of optimal for simple conjunctive target concepts (Corollary 5.7). Sections 6 and 7 extend these results to k-DNF, k-CNF and internal disjunctive target concepts (see Section 1). The main results are given in Corollaries 6.1 and 7.2 respectively. Finally, a number of remaining open problems are outlined in the conclusion.

Notation

We use "log" to denote the logarithm base 2 and "ln" to denote the natural logarithm. For any set S, $|S|$ denotes the cardinality of S.

1. Instances and Concepts

In the simplest type of inductive concept learning, each instance of a concept is defined by the values of a fixed set of *attributes*, not all of which are necessarily relevant. For example, an instance of the concept "red triangle" might be characterized by the fact that its color is red, its shape is triangular and its size is 5. Following [24], we consider three type of attributes. A *nominal* attribute is one that takes on a finite, unordered set of mutually exclusive values, e.g. the attribute *color*, restricted to the six primary and secondary colors, or a Boolean attribute, taking only the values *true* and *false*. A *linear* attribute is one with a linearly ordered set of mutually exclusive values, e.g. a real-valued or integervalued attribute. A *tree-structured* attribute is one with a finite set of hierarchically ordered values, e.g. the attribute *shape* shown in Fig. 1. Only the leaf values of a tree-structured attribute (e.g. the values *triangle*, *square hexagon*, *proper_ellipse*, *circle*, *crescent* and *channel* of Fig. 1) are directly observable. Since a nominal attribute can be converted to a tree-structured attribute by addition of the special value *any_value*, we will restrict our discussion to tree-structured and linear attributes. Throughout the paper we will assume that each attribute has at least two distinct observable values.

Let $A_1, \ldots, A_n$ be attributes with observable value sets $V_1, \ldots, V_n$ respectively, i.e. if A_i is a linear attribute then V_i contains all values of A_i and if A_i is tree-structured then V_i contains only the leaf values. The *instance space defined by* $A_1, \ldots, A_n$ is the cross-product of the value sets $V_1, \ldots, V_n$. Each instance in this space is characterized by an n-tuple giving an observable value for each attribute. The instance space can be thought of as consisting of a large set of simple objects, each object characterized by its properties as given by the values of attributes $A_1, \ldots, A_n$. Such an instance space is called *attribute-based*.[1]

Concepts can be specified on an instance space using a concept description language as described in [24]. Equations relating attributes to values will be called *atoms*, which are either *elementary* or *compound*. The possible forms of elementary atoms are as follows.

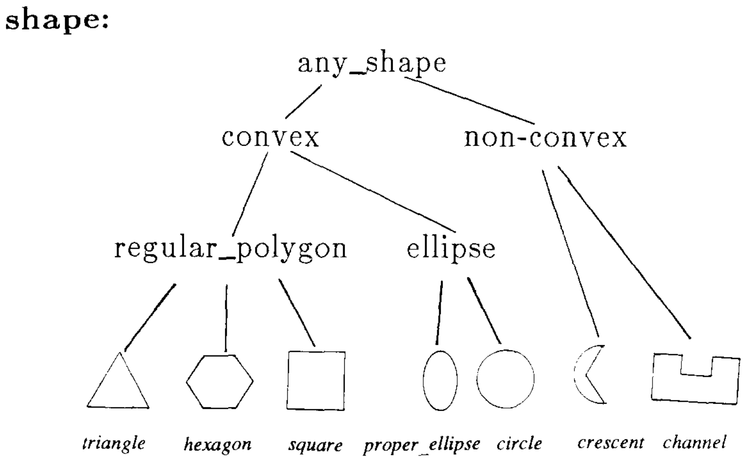


Fig. 1. The tree-structured attribute *shape*.

[1] A richer class of instance spaces, called *structured* instance spaces, can be defined by allowing each instance to include several objects, each with its own attributes, and allowing binary relations that define a structure between objects (see e.g. [10, 24]). The technique defined below for quantifying inductive bias and evaluating learning performance is extended to such spaces in [15].

– For tree-structured attributes:

attribute = *value* ,

e.g. *color* = *red*, *shape* = *regular_polygon*.

– For linear attributes:

$value_1 \leq attribute \leq value_2$,

e.g. $5 \leq size \leq 12$. Strict inequalities are also permitted, as well as intervals unbounded on one side. Atoms such as $5 \leq size \leq 5$ are abbreviated as $size = 5$.

Compound atoms can take the following forms.

– For tree-structured attributes:

attribute = $value_1$ *or* $value_2$ *or* . . . *or* $value_k$,

e.g. *shape* = *square or circle*.

– For linear attributes: any disjunction of intervals e.g. $0 \leq age \leq 21$ *or* $age \geq 65$. Disjunctive operators within compound atoms are called *internal disjunctions*.

We consider the following types of concepts:

(1) *Pure conjunctive*. Expressions are of the form

$atom_1$ *and* $atom_2$ *and* . . . *and* $atom_s$,

where each $atom_i$ is an elementary atom, $1 \leq i \leq s$. For example, *color* = *red and* $5 \leq size \leq 12$ is a pure conjunctive concept.

(2) *Pure disjunctive*. The same as pure conjunctive, but the atoms are connected by "*or*."

(3) *Internal disjunctive*. The same as pure conjunctive, but compound atoms are allowed. For example,

(*color* = *red or blue or yellow*) *and* (($5 \leq size \leq 12$) *or* ($size > 100$))

is an internal disjunctive concept.

(4) *k-DNF*. Expressions are of the form

t_1 *or* t_2 *or* . . . *or* t_s ,

where each t_i is a pure conjunctive concept with at most k atoms for some fixed k. For example,

(*color* = *red and shape* = *regular_polygon*)
or ($5 \leq size \leq 12$ *and shape* = *circle*)

is a 2-DNF concept. Within k-DNF concepts the pure conjunctive t_i are called *terms*.

(5) *k-CNF*. Expressions are of the form

$$c_1 \textit{ and } c_2 \textit{ and } \ldots \textit{ and } c_s\,,$$

where each c_i is a pure disjunctive concept with at most k atoms for some fixed k. The c_i are called *clauses*.

A concept of any of the above types represents a set of instances in the instance space in the usual way, i.e. the concept

$$\textit{shape} = \textit{regular\_polygon and } 5 \leqslant \textit{size} \leqslant 12$$

represents the set of all instances that have a value between 5 and 12 for the attribute *size* and a value for the attribute *shape* that is a leaf in the hierarchy below *regular_polygon*, i.e. *triangle*, *hexagon* or *square*. In what follows, we will not distinguish between the syntactic form of a concept (its *intension*) and the set of instances it represents (its *extension*), unless this distinction is required for clarity. We use the notation $x \in h$, the phrase "x is included in h" and the phrase "h covers x" interchangeably to denote that the instance x is an instance of the concept h.

2. Exhausting a Version Space

Let X be an instance space determined by a fixed set of attributes (each tree-structured or linear) and let H be a *hypothesis space* defined on X, i.e. a class of concepts defined using the attributes of X. For example, H might be the class of pure conjunctive concepts over X. Let Q be a finite set of examples of a target concept c defined on X. The *version space of Q (w.r.t. H)* is defined by Mitchell [27] as the set of all hypotheses in H that are consistent with all examples in Q.

Assume the instance space is finite. Then if the target concept c is a member of H, as new examples of c are added to Q, the version space of Q w.r.t. H shrinks until it eventually contains only the target concept c. If the target concept c is *not* a member of H, then as new examples of c are added to Q, the version space of Q w.r.t. H shrinks until it eventually becomes empty. We denote the fact that the version space has reached one of these two limiting states, i.e. is either empty or reduced to just the target concept, by saying that the version space is *exhausted (w.r.t. c)*.

Note that if the version space is reduced to one hypothesis h, but h is not the target concept, then the version space is not yet exhausted. This case can occur when the target concept is not a member of the hypothesis space H. In this case it is always possible to add a new example that distinguishes the hypothesis h from the target concept. This eliminates h from the version space, leaving it empty.

We say that the hypothesis h is more specific than the hypothesis h' if h is contained in h', and that h is more *general* than h' if h' is contained in h. A hypothesis h in the version space of Q is *maximally specific* if there is no other hypothesis h' in the version space of Q that is strictly more specific than h. *Maximally general* is defined similarly. Mitchell observes that by keeping track of only the maximally specific and the maximally general hypotheses in the version space of Q (the sets S and G respectively of [27]), we can monitor this version space as more examples are added to Q, and, while we cannot in general determine when it is exhausted, we can detect when it is either empty or has been reduced to just one hypothesis.[2] If it becomes empty, then we know that the target concept is not in the hypothesis space H. If it is reduced to just one hypothesis h, then we know that if the target concept is in the hypothesis space at all, then it must be h. This is sufficient for most learning applications.

There are two problems with this approach in practice. The first is that it may require too many examples to reduce the version space to at most one hypothesis. Consider the simple case when X is the instance space defined by the Boolean attributes $A_1, \ldots, A_n$, H is the class of pure conjunctive concepts over X and the target $c \in H$ is the concept $A_1 = true$. It is possible to observe all the 2^{n-2} positive examples of this concept in which $A_1 = true$ and (by coincidence) $A_2 = true$ as well, and all the 2^{n-2} negative examples of this concept in which $A_1 = false$ and $A_2 = false$, and still not be able to distinguish between the target concept $A_1 = true$ and the other consistent hypotheses $A_2 = true$, and $(A_1 = true)$ *and* $(A_2 = true)$. While it seems "unlikely" that such a sequence of examples will be given, if indeed the real target concept is $A_1 = true$, this intuition has not yet been quantified. Worse yet, if we use real-valued attributes and atoms that denote intervals of values on these attributes, then the version space can never be reduced to at most one hypothesis by any finite set of examples of any target concept.

The other problem with the version space approach (in Mitchell's model) is that even if we monitor only the sets S and G, the storage needed can still become exponentially large as we build up examples before it starts to drop as the version space approaches its limit. Bundy et al. [7] have noted that if X is defined by a finite set of tree-structured attributes and H is the class of pure conjunctive concepts over X, then the set S of maximally specific hypotheses in H that are consistent with a sample Q never contains more than one hypothesis. This holds for our more general notion of pure conjunctive concepts as well, as is demonstrated in Section 4 below. However, Bundy et al. fail to note that the set G of maximally general consistent hypotheses can grow exponentially large in the number of examples. This is demonstrated as follows.

[2] Other search techniques for version spaces are given in [22] in a more general setting.

As above, let X be defined by Boolean attributes $A_1, \ldots, A_n$ and H be the class of pure conjunctive concepts. Assume the number n of attributes is even. Let Q be the positive example

$$(\mathit{true}, \mathit{true}, \ldots, \mathit{true})$$

(i.e. $A_1, \ldots, A_n$ all have the value *true*) followed by the $\frac{1}{2}n$ negative examples

$$\begin{array}{l}(\mathit{false}, \mathit{false}, \mathit{true}, \mathit{true}, \mathit{true}, \ldots, \mathit{true}, \mathit{true}, \mathit{true})\,,\\ (\mathit{true}, \mathit{true}, \mathit{false}, \mathit{false}, \mathit{true}, \ldots, \mathit{true}, \mathit{true}, \mathit{true})\,,\\ \vdots\\ (\mathit{true}, \mathit{true}, \mathit{true}, \mathit{true}, \mathit{true}, \ldots, \mathit{true}, \mathit{false}, \mathit{false})\,.\end{array}$$

Assume h is a pure conjunctive hypothesis consistent with Q. In order to contain the positive example,

(1) h must be of the form

$$(A_{i_1} = \mathit{true})\ \mathit{and}\ (A_{i_2} = \mathit{true})\ \mathit{and}\ \ldots\ \mathit{and}\ (A_{i_k} = \mathit{true})$$

for some $\{A_{i_1}, \ldots, A_{i_k}\} \subseteq \{A_1, \ldots, A_n\}$. In other words, h cannot include an atom of the form $(A_i = \mathit{false})$. Given this restriction, to avoid containing a negative example,

(2) h must contain the following atoms:

either the atom $(A_1 = \mathit{true})$ or the atom $(A_2 = \mathit{true})$ and
either the atom $(A_3 = \mathit{true})$ or the atom $(A_4 = \mathit{true})$ and
$\vdots$
either the atom $(A_{n-1} = \mathit{true})$ or the atom $(A_n = \mathit{true})$.

The maximally general concepts that meet criteria (1) and (2) are those with the fewest atoms. It is easy to see that there are $2^{n/2}$ of these, all incomparable, each created by choosing one atom from each pair according to criterion (2). Hence, G has size exponential in the size of Q in this case.

These problems with the version space approach are overcome by incorporating into it the probabilistic ideas of the Valiant framework [39]. To overcome the first problem, we will abandon the goal of completely exhausting the version space and settle for a version space that is "probably almost exhausted" (cf. Dana Angluin's characterization of the Valiant framework as a "probably approximately correct" identification of a concept [1]). We will see below how this reduces the number of examples needed.

To overcome the second problem, we will simply avoid keeping track of the exact version space in any form. Instead, we will set things up so that any hypothesis from an "almost exhausted" version space will accurately approximate the target concept. Thus, the strategy of keeping track of all consistent hypotheses is replaced by the simpler strategy of drawing enough examples to probably almost exhaust the version space and then finding at least one hypothesis consistent with these examples.

We will assume that there is a fixed but arbitrary probability distribution defined on the instance space, unknown to the learner. This distribution can be as complex as it needs to be to adequately represent the real-world probabilities in any application domain. The complexity of the distribution will not affect the sample size bounds obtained below.

As outlined in the introduction, the notion of the *error* of a hypothesis with respect to the target concept is defined relative to this distribution: it is the combined probability of all instances that are either in the hypothesis and not in the target concept or in the target concept and not in the hypothesis, i.e. the probability of drawing a random example on which the hypothesis and target concept disagree. When the error is small, the hypothesis and the target concept differ only by a set of instances that rarely occur, i.e. the hypothesis is a good approximation to the target concept relative to the fixed "real-world" distribution on instances.

The idea of "almost exhausted" can now be formalized as follows.

Definition 2.1. Given a hypothesis space H, a target concept c, a sequence of examples Q of c, and an *error tolerance* ε, where $0 \leq \varepsilon \leq 1$, the version space of Q (w.r.t. H) is *ε-exhausted* (w.r.t. c) if it does not contain any hypothesis that has error more than ε with respect to c.[3]

Note that if the instance space is finite and no instance has zero probability, then setting $\varepsilon = 0$ in the above definition is equivalent to demanding that no hypothesis in the version space differ at all from the target concept, and thus this reduces to our original definition of an exhausted version space. Note also that since every hypothesis in an ε-exhausted version space has error at most ε with respect to the target concept, then any two hypotheses can have error at most 2ε with respect to each other, i.e. they will agree with each other except on a set of instances that has combined probability at most 2ε. Hence, when ε is small and the target concept is in H, although the version space may not be reduced to a single hypothesis, it is at least reduced to a set of hypotheses that are all almost identical to each other and to the target concept (with respect to the fixed probability distribution on the instance space).

How may examples are required to ε-exhaust a version space? As above, if we take the worst case over all possible sequences of distinct examples, then this number can be exponential or even infinite. The situation is considerably improved if we assume that the examples are drawn independently at random,

[3]The idea of ε-exhausting a version space is a special case of the more general idea of finding an ε-net for a set of regions, introduced in [17].

and insist only that the version space be ε-exhausted with high probability (hence the term "probably almost exhausted").

Lemma 2.2 [4, 42]. *If the hypothesis space H is finite, its cardinality denoted by $|H|$, and Q is a sequence of $m \geq 1$ independent random examples (chosen according to any fixed probability distribution on the instance space) of any target concept c, then for any $0 < \varepsilon < 1$, the probability that the version space of Q (w.r.t. H) is not ε-exhausted (w.r.t. c) is less than*

$$|H|e^{-\varepsilon m} .$$

Proof. Let $h_1, \ldots, h_k$ be the hypotheses in H that have error greater than ε with respect to c. We fail to ε-exhaust the version space w.r.t. H if and only if there is a hypothesis in this set that is consistent with all m independent random examples. Since each hypothesis has error greater than ε, an individual example of c is consistent with a given h_i with probability at most $(1 - \varepsilon)$. Thus, m independent random examples are consistent with h_i with probability at most $(1 - \varepsilon)^m$. Since the probability of a union of events is at most the sum of their individual probabilities, the probability that all m examples of c are consistent with any of the hypotheses in $h_1, \ldots, h_k$ is at most $k(1 - \varepsilon)^m$. The result follows from the fact that $k \leq |H|$ and $(1 - \varepsilon)^m \leq e^{-\varepsilon m}$ for $0 \leq \varepsilon \leq 1$ and $m \geq 0$. □

As a corollary of the above result, for any $\delta, 0 < \delta < 1$, if Q has size

$$m \geq (\ln(1/\delta) + \ln|H|)/\varepsilon , \tag{1}$$

then the version space of Q is ε-exhausted with probability at least $1 - \delta$. This follows from setting $|H|e^{-\varepsilon m} = \delta$ and solving for m. What is significant about this formula is that the number of random examples needed to ε-exhaust a version space is logarithmic in the size of the underlying hypothesis space, independent of the target concept and independent of the distribution on the instance space.

Compare this to the number of queries needed to (completely) exhaust the version space using the standard (nonprobabilistic) model. By a query we mean a question of the form "is x an instance of the target concept?," where x is any instance chosen by the learning algorithm. The minimum number of queries needed in the worst case to reduce a version space over a finite hypothesis space H to at most one hypothesis is $\log|H|$. This is achieved if there is a strategy that always cuts the version space in half with each new query [27]. For fixed ε and δ this is of the same order of magnitude as the bound given in equation (1) above.

Returning to our example of the instance space defined by n Boolean attributes $A_1, \ldots, A_n$, if H is the hypothesis space of all pure conjunctive concepts over this instance space, then $|H| = 3^n$ (for each attribute A we can include the atom $A = true$, the atom $A = false$ or neither), hence the version space will be ε-exhausted with probability $1 - \delta$ after

$$(\ln(1/\delta) + n \ln 3)/\varepsilon$$

independent random examples, regardless of the underlying distribution governing the generation of these examples. Note that the number of examples required grows only linearly in the number n of attributes, instead of exponentially in n as it does for completely exhausting the version space.

Upper estimates on the number of examples needed to ε-exhaust a version space that are derived by the above method are still very crude, and for the case of infinite hypothesis spaces, such as the set of intervals on the real line, the method does not even apply. We remedy this in the next section.

3. The Growth Function and the Vapnik–Chervonenkis Dimension

We use the following notions from [42, 43] (see also [9]).

Definition 3.1. Let X be an instance space and H be a hypothesis space defined on X. Let I be a finite set of instances in X. For a given hypothesis $h \in H$, label I so that it becomes a sample of h, i.e. label all the instances of I included in h with "+" and the others with "−". This labeling partitions I into a set of positive instances and a set of negative instances. This partition is called the *dichotomy of I induced by h*. $\Pi_H(I)$ denotes the set of all dichotomies of I induced by hypotheses in H, i.e. the set of all ways the instances in I can be labeled with "+" and "−" so as to be consistent with at least one hypothesis in H. For any integer m, $1 \leq m \leq |X|$, $\Pi_H(m) = \max|\Pi_H(I)|$ over all sets of instances $I \subseteq X$ such that $|I| = m$. Hence, $\Pi_H(m)$ is the maximum number of dichotomies induced by hypotheses in H on any set of m instances. As in [42], we call $\Pi_H(m)$ the *growth function* of H.

As an example, let X be the instance space defined by the tree-structured attribute *shape* given in Fig. 1 and let H be the hypothesis space of all pure conjunctive concepts on X. Since X is defined by a single tree-structured attribute, any conjunction in H can be reduced to a single atom, and hence the hypotheses in H are given by the nodes in the hierarchy depicted in Fig. 1.

Let $I = \{tri, sq, cir\}$ be a set of three instances in X, where *tri* is an instance with *shape = triangle*, *sq* an instance with *shape = square*, and *cir* an instance with *shape = circle*. Then the hypothesis *shape = regular_polygon* induces the dichotomy $\{(tri, +), (sq, +), (cir, -)\}$ of I. The hypothesis *shape = ellipse* induces the complementary dichotomy $\{(tri, -), (sq, -), (cir, +)\}$ of I. The hypotheses *shape* = *triangle*, *shape* = *square*, *shape* = *convex* and *shape* = *non_convex* induce four more distinct dichotomies on I, for a total of six dichotomies. However, there is no hypothesis that induces the dichotomy $\{(tri, +), (sq, -), (cir, +)\}$ of I. Because the least common ancestor of *triangle* and *circle* in the *shape* hierarchy is *convex*, which already includes *square*, the concept description language of H cannot represent a hypothesis that includes triangles and circles but not squares. The same is true for the dichotomy $\{(tri, -), (sq, +), (cir, +)\}$. Hence, in this case $|\Pi_H(I)| = 6$. This implies that $\Pi_H(3) \geq 6$, since $\Pi_H(m)$ is the maximum of $|\Pi_H(I)|$ over all sets I of m instances.

In fact $\Pi_H(3) = 6$ in this case, since it is easily verified that $|\Pi_H(I)| \leq 6$ for any set I of 3 distinct instances whenever X is defined by a single tree-structured attribute and the hypothesis space H is pure conjunctive. Ultimately, this follows from the fact that for any 3 leaves of a tree, 2 of them always have a least common ancestor that is either equal to or a descendant of the least common ancestor of all 3 leaves. Since not all of the 8 possible dichotomies of 3 instances can be expressed, this represents a kind of bias inherent in the hypothesis space H, which may be attributed to its restricted concept description language. This bias is not evident when we consider sets I containing only 2 instances. Even in the *shape* example, for any such set all 4 dichotomies are induced by hypotheses in H. Hence, $\Pi_H(2) = 4 = 2^{|I|}$, its maximum possible value.

Definition 3.2. Let I be a set of instances in X. If H induces all possible $2^{|I|}$ dichotomies of I, then we say that H *shatters* I. The *Vapnik–Chervonenkis dimension* of H, denoted VCdim(H), is the cardinality of the largest finite subset I of X that is shattered by H, or equivalently, the largest m such that $\Pi_H(m) = 2^m$. If arbitrarily large subsets of X can be shattered, then VCdim(H) = ∞.

Continuing with the example given above, since $\Pi_H(2) = 4$ and $\Pi_H(3) = 6 < 8$, VCdim(H) = 2. (Note that by the definition of Π_H, if $\Pi_H(m_0) < 2^{m_0}$ then $\Pi_H(m) < 2^m$ for all $m \geq m_0$.) In general, VCdim(H) ≤ 2 whenever the instance space X is defined by a single tree-structured attribute and the hypothesis space H is pure conjunctive. In a similar manner, it is easily verified that whenever X is defined by a single linear attribute, say *size*, and the hypothesis space H is pure conjunctive, then VCdim(H) ≤ 2 as well. In this case, the hypothesis space H can be represented by all possible size intervals. For any 3 instances with distinct sizes $x < y < z$, there is no size interval that includes the instances with sizes x and z without also including the instance with size y. Thus no set $I \subseteq X$ of cardinality 3 is shattered by H. Note that this holds even when *size* is real-valued, and hence the cardinalities of X and H are infinite. Note also that in the linear case $\Pi_H(3)$ is 7 instead of 6.

The following result, derived from the pioneering work in [42, 43], is a natural analogue to Lemma 2.2 of the previous section. It relates the growth function $\Pi_H(m)$ to the number of examples required to ε-exhaust a version space with respect to H.

Theorem 3.3[4] ([5, Theorem A2.4]. See also [17]). *If H is a hypothesis space and Q is a sequence of $m \geq 1$ independent random examples (chosen according to any fixed probability distribution on the instance space) of any target concept c, then for any $0 < \varepsilon < 1$, the probability that the version space of Q (w.r.t. H) is not ε-exhausted (w.r.t. c) is less than*

$$2\Pi_H(2m)2^{-\varepsilon m/2} .$$

The following bounds on the growth function in terms of VCdim(H) are given in [5, Proposition A2.5] (also derived from [42]).

Lemma 3.4. *If* $\mathrm{VCdim}(H) = d$ *and* $m \geq d \geq 1$, *then* $\Pi_H(m) \leq (\mathrm{e}m/d)^d$, *where* e *is the base of the natural logarithm.*

As in the previous section, using Lemma 3.4 we can set the value given in Theorem 3.3 to δ and "solve" for m. From the calculation given in [5, Lemma A2.6] we have the following:

Corollary 3.5. *If the sample Q has size at least*

$$(4\log(2/\delta) + 8\,\mathrm{VCdim}(H)\log(13/\varepsilon))/\varepsilon ,$$

then the version space of Q (w.r.t. H) is ε-exhausted with probability at least $1 - \delta$.

Let us compare these results to the analogous results from the previous section. Assume the hypothesis space H is finite and $\mathrm{VCdim}(H) = d$. Hence, there exists a set I of d distinct instances that is shattered by H. Since this requires 2^d distinct hypotheses, $|H| \geq 2^d$. Therefore $d = \mathrm{VCdim}(H) \leq \log|H|$ whenever H is finite. In the above example of pure conjunctive hypotheses on a single tree-structured or linear attribute $\mathrm{VCdim}(H) \leq 2$, but $\log|H|$ can be arbitrarily large. This shows that in many cases VCdim(H) is much less than $\log|H|$. This often happens when the hypothesis space has some special structure that weakens its "power of expression" and thereby holds its growth function down. In these cases Corollary 3.5 can be significantly better than the corollary to Lemma 2.2 given in equation (1), despite the larger constants and the additional $\log(13/\varepsilon)$ factor.

[4] Here and in subsequent results we are suppressing some additional measurability assumptions required in the general form of the theorem since they will not be relevant in our intended applications (see [5, Appendix]).

On the other hand, if H is finite and VCdim(H) is not significantly smaller than $\log|H|$, then instead of using the bound on $\Pi_H(m)$ given in Lemma 3.4, we can simply use the bound $\Pi_H(m) \leq |H|$. This follows from the fact that each dichotomy must be induced by a distinct hypothesis in H. Using this bound, setting the value given in Theorem 3.3 to δ and solving for m gives results similar to those given in equation (1), but with slightly higher constants.

Because it reflects limitations on the power of discrimination and expression inherent in the hypothesis space H, the growth function $\Pi_H(m)$ is a natural way to quantify the bias of learning algorithms that use H. It is also a useful measure of bias. Theorem 3.3 provides a direct way to use this measure of bias to determine how fast a version space with respect to H shrinks, in a probabilistic sense. In subsequent sections we will see how this result translates into performance bounds on learning algorithms that use the hypothesis space H.

Lemma 3.4 shows that $\Pi_H(m)$ grows as 2^m until m reaches a critical value $d = \text{VCdim}(H)$, and thereafter grows polynomially in m, with exponent at most d. Beyond this critical value, the polynomial growth function $\Pi_H(m)$ is rapidly dominated by the negative exponential $2^{-\varepsilon m/2}$ in the formula of Theorem 3.3. Because of this, many useful learning performance bounds can be obtained directly from VCdim(H), without considering other details of the growth function. In some cases this is also true of $\log|H|$, which we have seen is an upper bound on VCdim(H). Hence, these values are also useful measures of bias.

We now give bounds on the growth function and VC dimension of each of the more general concept classes introduced in Section 1. These results are derived in part from results in [23, 44]. The reader anxious to forge ahead to learning applications can safely skip the proof of the following theorem without loss of continuity.

Theorem 3.6. *Let X be an instance space defined by $n \geq 1$ attributes, each tree-structured or linear.*

(i) *If H is the hypothesis space of all pure conjunctive concepts on X, then*

$$n \leq \text{VCdim}(H) \leq 2n$$

and

$$\Pi_H(m) \leq (\tfrac{1}{2}em/n)^{2n} \quad \textit{for all } m \geq 2n .$$

(ii) *If H is the hypothesis space of*

(a) *all pure conjunctive concepts on X that contain at most s atoms,*

(b) *all pure disjunctive concepts on X that contain at most s atoms, or*

(c) *all internal disjunctive concepts with at most s occurrences of tree-structured attribute values or linear attribute value ranges in all compound atoms combined, then*

$$\Pi_H(m) \leq n^s m^{2s} \quad \textit{for all } m \geq 2,$$

$$\mathrm{VCdim}(H) \leq 4s \log(4s\sqrt{n}),$$

and

$$s\lfloor \log(n/s) \rfloor \leq \mathrm{VCdim}(H) \quad \textit{for } s \leq n.$$

(iii) *If H is the hypothesis space of all k-DNF concepts on X with at most s terms* (*or k-CNF concepts with at most s clauses*), *then*

$$\Pi_H(m) \leq n^{ks} m^{2ks} \quad \textit{for all } m \geq 2,$$

$$\mathrm{VCdim}(H) \leq 4ks \log(4ks\sqrt{n}),$$

and

$$ks\left\lfloor \log \frac{n}{ks^{1/k}} \right\rfloor \leq \mathrm{VCdim}(H) \quad \textit{for } k \leq n \textit{ and } s \leq \binom{n}{k}.$$

Proof. The proof of this result is given in a series of lemmas.

Lemma 3.7 [44]. *If X is an instance space defined by n linear attributes $A_1, \ldots, A_n$ and H is the set of pure conjunctive concepts over X, then* $\mathrm{VCdim}(H) \leq 2n$.

Proof. Recall that instances in X are represented as n-tuples of values over $A_1, \ldots, A_n$. Let I be a subset of X of cardinality $2n+1$. For each i, $1 \leq i \leq n$, choose a member of I that has the largest value for the attribute A_i among all members of I, and a member of I that has the smallest value for the attribute A_i among all members of I. Let S be the set of all members of I that are chosen. Elements of S will be called *extreme* members of I (see Fig. 2). Clearly I can have at most $2n$ extreme members, and thus I has at least one element that is not extreme. Furthermore, since the hypotheses of H are cross-products of intervals of values of the attributes $A_1, \ldots, A_n$, it is easily verified that any

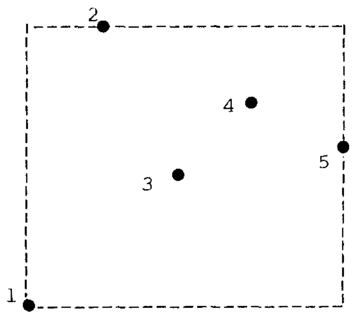


Fig. 2. Case $n = 2$, cardinality of I is 5. Extreme points are 1, 2 and 5. Any pure conjunctive hypothesis that contains all extreme points must contain the dashed region, hence the points 3 and 4.

hypothesis that includes all extreme members of I includes all members of I. Hence, if we form a dichotomy of I by labeling all extreme members of I as positive instances and all other members of I as negative instances, then no hypothesis in H is consistent with this labeling. Thus I is not shattered by H. It follows that no subset of cardinality $2n + 1$ can be shattered by H, showing that the VC dimension of H is at most $2n$. □

It is shown in [44] that $\mathrm{VCdim}(H) = 2n$ when H is the space of pure conjunctive concepts on n real-valued linear attributes. Hence, this upper bound cannot be improved.

Corollary 3.8. *If X is an instance space defined by n attributes, each linear or tree-structured, and H is the set of pure conjunctive concepts over X, then* $\mathrm{VCdim}(H) \leq 2n$.

Proof. The observed values (leaves) of any tree-structured attribute can be ordered in such a way that any higher-level value (internal node) represents an interval of observed values. Hence, H is a subset of some H', where H' is the class of pure conjunctive concepts over some set of n linear attributes. Since the VC dimension of a subclass of concepts is never more than the VC dimension of the class itself, by the previous lemma, this implies that the VC dimension of H is at most $2n$. □

This establishes the upper bound on $\mathrm{VCdim}(H)$ in this case. For the lower bound we will assume that the instance space X is defined by n Boolean attributes $A_1, \ldots, A_n$. Since we are assuming throughout the paper that each attribute has at least two observable values, we can make this assumption without loss of generality. Let I be the set of instances

$$\begin{aligned}
x_1 &= (\mathit{false}, \mathit{true}, \mathit{true}, \ldots, \mathit{true})\,, \\
x_2 &= (\mathit{true}, \mathit{false}, \mathit{true}, \ldots, \mathit{true})\,, \\
x_3 &= (\mathit{true}, \mathit{true}, \mathit{false}, \ldots, \mathit{true})\,, \\
&\vdots \\
x_n &= (\mathit{true}, \mathit{true}, \mathit{true}, \ldots, \mathit{false})\,.
\end{aligned}$$

It is easily verified that for any $\{i_1, i_2, \ldots, i_k\} \subseteq \{1, \ldots, n\}$, the dichotomy of I in which all instances $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ are labeled "$-$" and all others are labeled "+" is induced by the pure conjunctive concept

$$(A_{i_1} = \mathit{true}) \text{ and } (A_{i_2} = \mathit{true}) \text{ and } \ldots \text{ and } (A_{i_k} = \mathit{true})\,.$$

Hence, I is shattered by H and thus $\mathrm{VCdim}(H) \geq n$.

Now note that by Lemma 3.4, if the VC dimension of H is $2n$, then

$\Pi_H(m) \leq (\frac{1}{2}em/n)^{2n}$ for all $m \geq 2n$. This also holds for smaller VC dimensions, since for all k, any hypothesis space with VC dimension less than k is contained in a hypothesis space with VC dimension equal to k. This, combined with the above results, establishes Theorem 3.6(i).

Lemma 3.9. *If H is the hypothesis space of*

(a) *all pure conjunctive concepts on X that contain at most s atoms,*

(b) *all pure disjunctive concepts on X that contain at most s atoms, or*

(c) *all internal disjunctive concepts with at most s occurrences of tree-structured attribute values or linear attribute value ranges in all compound atoms combined, then*

$$\Pi_H(m) \leq n^s m^{2s} \quad \textit{for all } m \geq 2\,.$$

Proof. Let I be a set of $m \geq 2$ instances in X. We first claim that for any (elementary) atom involving a linear attribute A, there are at most $\binom{m}{2} + m + 1$ ways this atom can induce a dichotomy on the set I by partitioning it into positive instances whose values on A satisfy the atom and negative instances whose values do not. To see this, order the elements of I as $x_1, \ldots, x_m$ such that for each i, $1 \leq i < m$, the value of A on x_i is less than or equal to the value of A on x_{i+1}. Since each atom involving the attribute A specifies an interval of values of A, each such atom induces a dichotomy on I by making positive some interval of instances $x_i, \ldots, x_j$, where $1 \leq i \leq j \leq m$, and making the rest negative, or by making all instances negative. This gives at most $\binom{m}{2} + m + 1$ dichotomies.

As in the previous lemma, since the leaves of any tree can be ordered so that the set of leaves of the subtree defined by any internal node forms an interval of this ordering, this result also holds for tree-structured attributes. (A tighter bound of at most $2m$ dichotomies can also be derived for the tree-structured case.)

It is easily verified that $\binom{m}{2} + m + 1 \leq m^2$ for all $m \geq 2$. Hence, we have shown that for each attribute A, the atoms involving A are capable of inducing at most m^2 dichotomies on a set I of m instances. The dichotomy induced by a hypothesis formed by the conjunction or disjunction of a set of atoms is entirely determined by the dichotomies induced by the individual atoms. Since it does not change the hypothesis to include the same atom more than once, we can assume without loss of generality that each hypothesis $h \in H$ contains exactly s atoms. Since for each of the s atoms in the hypothesis h there are n ways to assign it an attribute and at most m^2 ways to choose the dichotomy induced by its value range given its assigned attribute, this gives a bound of $(nm^2)^s = n^s m^{2s}$ on the total number of distinct dichotomies induced by H on I. Hence, $\Pi_H(m) \leq n^2 m^{2s}$ in cases (a) and (b).

Clearly the same argument works in case (c) for internal disjunctive concepts. Once we have assigned attributes to elementary atoms, we can collect all the elementary atoms that share a common attribute together to form compound atoms and then form the conjunction of these. Every internal disjunctive concept can be formed in this way. The dichotomy it induces is determined by the dichotomies of the elementary atoms and the way attributes are assigned to them. □

Lemma 3.10. *If H is the hypothesis space of all k-DNF concepts on X with at most s terms (or k-CNF concepts with at most s clauses), then*

$$\Pi_H(m) \leq n^{ks} m^{2ks} \quad \textit{for all } m \geq 2\,.$$

Proof. By Lemma 3.9, the number of dichotomies induced by a single term of a k-DNF is at most $n^k m^{2k}$. As above, we can assume that the k-DNF expression contains exactly s terms. Since the dichotomies induced by a k-DNF expression are determined by the dichotomies induced by each of its terms, there are at most $(n^k m^{2k})^s = n^{ks} m^{2ks}$ dichotomies induced by k-DNF expressions with s terms. Clearly the same argument works for k-CNF. □

Lemma 3.11. *Assume $n,s \geq 1$. Then for any $m > 4s\log(4s\sqrt{n})$, $n^s m^{2s} < 2^m$.*

Proof. This is easily verified. □

The upper bounds in Theorem 3.6(ii) and (iii) follow directly from Lemmas 3.9–3.11. The lower bounds follow from [23, Lemma 4.6] (see also [23, Example 4 in Section 5]), which uses an example on an instance space of Boolean attributes remotely related to that given above for the lower bound in part (i). This completes the proof of Theorem 3.6. □

As an example application of Theorem 3.6, we can now extend the result obtained in the previous section for the hypothesis space of pure conjunctive concepts over an instance space of n Boolean attributes to pure conjunctive concepts over n arbitrary tree-structured and linear attributes. Since by Theorem 3.6(i) the Vapnik–Chervonenkis dimension of this hypothesis space is at most $2n$, using Corollary 3.5, after

$$(4\log(2/\delta) + 16n\log(13/\varepsilon))/\varepsilon\,,$$

independent random examples of any target concept c, the version space w.r.t. this hypothesis space will be ε-exhausted (w.r.t. c) with probability at least $1-\delta$, independent of the distribution governing the generation of the examples.

Note that this bound is not much higher than that given in Section 2 for the case of Boolean attributes. In particular, this bound does not depend on the size or complexity of the hierarchies of values defined for the tree-structured attributes, nor on the number of values of the linear attributes. In fact, the linear attributes can be real-valued. This is because increasing the number of values of the attributes does not increase the Vapnik–Chervonenkis dimension of the hypothesis space beyond $2n$, no matter how much it increases the size of the hypothesis space.

Similar bounds hold for the other kinds of hypothesis spaces treated in Theorem 3.6.

4. The Performance of the Classical Learning Algorithm for Conjunctive Concepts

The fact that the hypothesis space of pure conjunctive concepts is rapidly ε-exhausted as independent random examples of any target concept are drawn tells us a good deal about the performance of learning algorithms that use this hypothesis space. Here we apply this result to analyze the performance of one of the simplest learning algorithms for pure conjunctive concepts, which we will call the *classical algorithm*. To analyze learning performance we will adopt the viewpoint of Valiant [39] and ask how many random examples and how much computational effort is required for the algorithm to, with high probability, find a hypothesis that is a good approximation of the target concept.

Let X be a fixed instance space defined by n attributes, each tree-structured or linear. Let Q be a sample of any concept defined on X. For simplicity, we will assume here and in what follows that the sample Q contains at least one positive example. Under this assumption, for any attribute A the *minimal dominating atom for* A (*w.r.t.* Q) is defined as the most specific elementary atom involving the attribute A that includes all the positive examples of Q. It is easily verified that this atom is always uniquely defined for tree-structured and linear attributes. If A is a linear attribute, the minimal dominating atom for A is the atom $v_1 \leq A \leq v_2$, where v_1 and v_2 are the smallest and largest values of A that occur among the positive examples. This atom is the result of applying the "closing interval rule" of [24]. If A is a tree-structured attribute, the minimal dominating atom is $A = v$, where v is the value of the node that is the least common ancestor of all the leaf values of A that occur among the positive examples (see Fig. 3(b) for an example). This atom is the result of using the climbing tree heuristic of [24]. It also corresponds to the "lower mark" in the attribute trees of [7].

We can use the minimal dominating atoms to find the unique most specific pure conjunctive concept consistent with a given sample. This learning method can be traced back in various forms at least to [6]. It leads to the following:[5]

[5]This algorithm is typically presented as an incremental algorithm, but this causes problems with the negative examples [7, 27]. Therefore we give it in a non-incremental form.

Algorithm 4.1 (Classical algorithm for learning conjunctive concepts).

Step 1. Find the minimal dominating atom for each attribute with respect to the given sample. Let the conjunction of these atoms be the hypothesis h.

Step 2. If no negative examples are included in h then return h, else report that the sample is not consistent with any pure conjunctive concept.

To illustrate this algorithm, consider an instance space with attributes *shape*, *size* and *shade*, where *shape* is the tree-structured attribute given in Fig. 1, *size* is a real-valued linear attribute, and *shade* is Boolean. Let the sample Q consist of the positive examples

(*square*, 5.2, *true*),
(*triangle*, 3.4, *true*),
(*square*, 2.9, *true*),

and the negative examples

(*circle*, 4.3, *true*),
(*channel*, 5.1, *true*),
(*square*, 3.7, *false*).

Then the minimal dominating atoms are

$shape = regular\_polygon$,
$2.9 \leq size \leq 5.2$,
$shade = true$.

Hence, Algorithm 4.1 forms the conjunction of these as its hypothesis. No negative examples are included in this hypothesis, hence it is returned.

Lemma 4.2. *If there exists a pure conjunctive concept consistent with the sample, Algorithm* 4.1 *will find the unique maximally specific such concept, otherwise it correctly reports that the sample is not consistent with any pure conjunctive concept.*

Proof. Let $h = a_1$ *and* a_2 *and* . . . *and* a_n, where a_i is the minimal dominating atom for the attribute A_i w.r.t. Q. For each i, let V_i denote the set of values for A_i included in the atom a_i. The hypothesis h represents the set of all instances in the cross-product of $V_1, \ldots, V_n$. By the definition of a minimal dominating atom, for any positive example $(v_1, \ldots, v_n)$ we must have $v_i \in V_i$ for all i and hence this example is included in h. Thus if h does not include any negative examples, then it is consistent with Q. On the other hand, since each a_i is the unique minimal dominating atom for A_i, any other atom that includes all values of A_i that occur in positive examples must include all values in V_i. Therefore any conjunction of such atoms must represent a hypothesis that includes all examples in the cross product of $V_1, \ldots, V_n$. Therefore any pure conjunctive hypothesis that is consistent with the sample must contain h. It follows that if h does not include any negative examples, then h is the unique maximally specific pure conjunctive hypothesis that is consistent with Q, otherwise no pure conjunctive hypothesis is consistent with Q. □

In order to analyze the performance of this algorithm, let us first make the following general definition.

Definition 4.3. We say that a learning algorithm *uses the hypothesis space H consistently* if for any sequence of examples Q:

(1) if the version space of Q (w.r.t. H) is not empty, then the algorithm produces a hypothesis in this version space,

(2) else it indicates that no hypothesis in H is consistent with the given examples.

Lemma 4.2 shows that Algorithm 4.1 uses the hypothesis space of pure conjunctive concepts consistently. More sophisticated learning algorithms may handle case (2) more intelligently by "shifting the bias" when the version space becomes empty, as described in [38]. However, it is still likely that they will use procedures that act as described in (1) and (2) to detect the need to shift bias, so in general, the performance of such procedures still warrants investigation. In this regard, we have the following result.

Theorem 4.4. *Let H be a hypothesis space and L be a learning algorithm that uses H consistently. For any $0 < \varepsilon, \delta < 1$, given*

$$(4 \log(2/\delta) + 8\,\mathrm{VCdim}(H) \log(13/\varepsilon))/\varepsilon$$

independent random examples of any target concept c, with probability at least $1 - \delta$, algorithm L will either

(1) *produce a hypothesis in H that has error at most ε with respect to c, or*

(2) *indicate correctly that the target concept c is not in H.*

Moreover, this result holds regardless of the particular probability distribution on the instance space that governs the generation of examples.

(Note: we do not claim that whenever $c \not\in H$ the algorithm detects this with high probability. It may instead find a good approximation to c in H.)

Proof. By Corollary 3.5, after this many examples the version space with respect to H is ε-exhausted with probability $1 - \delta$. When the version space is ε-exhausted then either it is empty, in which case, since L uses H consistently,

L halts, indicating correctly that no hypothesis in H is consistent with the given examples of c and hence $c \not\in H$, or it is not empty, in which case L produces a hypothesis from this space and, because the space is ε-exhausted, this hypothesis has error at most ε. □

This gives the following result on the performance of the classical learning algorithm for pure conjunctive concepts.

Corollary 4.5. *Let X be an instance space defined by n attributes, each tree-structured or linear. For any $0 < \varepsilon, \delta < 1$, given*

$$(4 \log(2/\delta) + 16n \log(13/\varepsilon))/\varepsilon$$

independent random examples of any target concept c defined on X, with probability at least $1 - \delta$, Algorithm 4.1 will either

(1) *produce a pure conjunctive hypothesis that has error at most ε with respect to c, or*

(2) *indicate correctly that the target concept c is not a pure conjunctive concept.*

This holds for any probability distribution on X governing the generation of examples.

Proof. Lemma 4.2 shows that Algorithm 4.1 uses the hypothesis space H of pure conjunctive concepts on X consistently and Theorem 3.6(i) shows that $\mathrm{VCdim}(H) \leq 2n$. The result then follows directly from Theorem 4.4. □

This result shows that whenever the target concept is pure conjunctive, the classical learning algorithm will find a good approximation to it with high probability using relatively few random examples. The number of examples required is at most linear in the number of attributes in the instance space, almost linear in the inverse of the error parameter ε, and logarithmic in the inverse of the confidence parameter δ. One remarkable aspect of this result is that this bound on the number of examples required does not depend on the number of values that each attribute in the instance space has. As mentioned in the previous section, this is because all pure conjunctive hypothesis spaces on n tree-structured or linear attributes have VC dimension at most $2n$, regardless of the number of values per attribute.

How close does this upper estimate come to the actual number of examples needed for probably approximately correct learning? How does this number of examples compare to the number of examples needed by other algorithms? In order to answer these questions, we make the following definition.

Definition 4.6. Let L be a learning algorithm and C be a class of target concepts on the instance space X. For any $0 < \varepsilon, \delta < 1$, $S_C^L(\varepsilon, \delta)$ denotes the minimum sample size m such that for any target concept $c \in C$ and any distribution on X, given m random examples of c, L produces a hypothesis that, with probability at least $1 - \delta$, has error at most ε. $S_C^L(\varepsilon, \delta)$ is called the *sample complexity* of L for the target class C.

Theorem 4.7 [12]. *If C is a class of concepts with* $\mathrm{VCdim}(C) \geq 2$, *then there exists a positive constant c_0 such that for all learning algorithms L,*

$$S_C^L(\varepsilon, \delta) \geq c_0(\log(1/\delta) + \mathrm{VCdim}(C))/\varepsilon$$

for all sufficiently small[6] *positive ε and δ.*

Corollary 4.8. *There are positive constants c_0 and c_1 such that for any instance space X defined on n attributes, each tree-structured or linear*

$$c_0(\log(1/\delta) + n)/\varepsilon \leq S_C^L(\varepsilon, \delta) \leq c_1(\log(1/\delta) + n \log(1/\varepsilon))/\varepsilon$$

for all sufficiently small ε and δ, where L is Algorithm 4.1 *and C is the class of pure conjunctive concepts on X. Moreover, this lower bound holds for any learning algorithm L.*

Proof. Using the fact that $n \leq \mathrm{VCdim}(C)$ from Theorem 3.6(i), the first inequality follows from Theorem 4.7. The second inequality follows from Corollary 4.5. □

Corollary 4.8 shows that we have overestimated the sample complexity of Algorithm 4.1 by at most an $O(\log(1/\varepsilon))$ factor. More importantly, it shows that the actual sample complexity of Algorithm 4.1, whatever it is, is within an $O(\log(1/\varepsilon))$ factor of optimal for any learning algorithm for pure conjunctive concepts.

Algorithm 4.1 is also extremely efficient computationally. In order to analyze the time complexity of this algorithm, for simplicity we assume that for a linear attribute the time required to compare two values is constant, and for a tree-structured attribute the time required to determine if one value is in the subtree below another value or to compute the least common ancestor of two values is constant. This will not be an unreasonable approximation in most applications.

Under these assumptions the time required to find the minimal dominating atom for a single attribute with respect to a sample of size m is $O(m)$. Hence, the time for Step 1 of Algorithm 4.1 is $O(nm)$ on an instance space with n

[6] The result in [12] shows that this holds for all $\varepsilon < 1/8$ and $\delta \leq 1/100$.

attributes. Step 2 takes no longer, hence the overall time for Algorithm 4.1 is $O(nm)$. This is essentially optimal, since for the standard encoding of instances as n-tuples, the size of the sample itself is proportional to nm, and hence $\Omega(nm)$ time is required merely to read the sample.

5. Using a Greedy Heuristic to Improve Performance on Simpler Target Concepts

In many AI learning situations where conjunctive concepts are used, the task is to learn relatively simple conjuncts from examples over instance spaces with many attributes. This is because without a fairly strong domain theory, it is hard to anticipate in advance which attributes each individual target concept will depend on, and so a large number of possible attributes are considered for all target concepts. This problem becomes particularly acute in large scale systems in which each new learned concept is allowed to depend on previously learned concepts (viewed as Boolean attributes), and in systems where a large "library" of attributes is derived from simple combinations of primitive attributes [23, 35].

It is therefore of some interest to consider the problem of learning target concepts on an instance space defined by n attributes, where each target concept is represented by a pure conjunctive expression with at most s atoms, with s much smaller than n. If C is the class of all such target concepts, then Theorem 3.6(ii) shows that $\text{VCdim}(C) \leq 4s \log(4s\sqrt{n})$. Since this bound is logarithmic in n, when s is small relative to n it is considerably better than n, which is a lower bound on the VC dimension of the class of all pure conjunctive concepts on an n-attribute instance space. In view of Theorems 4.4 and 4.7, this indicates that it may be possible to learn concepts in C with considerably fewer random examples than are required to learn arbitrary pure conjunctive concepts on an n-attribute instance space.

This is indeed the case. Instead of using the classical algorithm, which finds the most specific conjunct that is consistent with the sample, consider an algorithm that finds the *simplest* conjunct, i.e. the conjunct with the least number of atoms, that is consistent with the sample. For now, let us assume that this is accomplished by an exhaustive search.

Given a sample of any target concept c in C, this algorithm always produces a conjunct that is consistent with the sample, and contains no more atoms than c itself. Hence, given any sample of a target concept in C, this algorithm will find a consistent hypothesis in C. If it cannot find a consistent hypothesis in C, then the target concept cannot be in C. Thus for any particular C, the algorithm can easily be adapted to use the hypothesis space C consistently. If L is the resulting algorithm, then by Theorem 4.3, using the bound from Theorem 3.6(ii) on $\text{VCdim}(C)$, we can show that

$$S_C^L(\varepsilon, \delta) \leq (4 \log(2/\delta) + 32s \log(4s\sqrt{n}) \log(13/\varepsilon))/\varepsilon . \qquad (2)$$

When s is very small and n very large, this sample complexity is considerably smaller than that given in Corollary 4.8 (with constants from Corollary 4.5) for the target class of all pure conjunctive concepts using the classical learning algorithm.

Of course this result is of limited value since exhaustively searching for the simplest consistent conjunct requires exponential time, and thus this learning algorithm is entirely impractical as it stands. Can this algorithm be efficiently implemented using a different method? The following shows that it probably cannot.

Theorem 5.1. *Given a sample on n attributes that is consistent with some pure conjunctive hypothesis, it is NP-hard to find a pure conjunctive hypothesis that is both consistent with this sample and has the minimum number of atoms.*

Proof. We will reduce the following problem, known to be NP-hard [13], to the above problem.

Minimum set cover problem. Given a collection of sets with union T (i.e. that cover T), find a subcollection whose union is T that has the minimum number of sets. This is called a *minimum cover* of T.

Given an instance of the minimum set cover problem defined by the collection of sets $S_1, \ldots, S_n$ with union $T = \{x_1, \ldots, x_k\}$, let $A_1, \ldots, A_n$ be a set of Boolean attributes. Let the sample Q consist of one positive example

$$(\mathit{true}, \mathit{true}, \ldots, \mathit{true})$$

followed by k negative examples

$$\begin{gathered}(v_{1,1}, v_{1,2}, \ldots, v_{1,n}), \\ \vdots \\ (v_{k,1}, v_{k,2}, \ldots, v_{k,n}),\end{gathered}$$

where for all i, $1 \leq i \leq k$ and all j, $1 \leq j \leq n$, $v_{i,j} = \mathit{false}$ if $x_i \in S_j$ and $v_{i,j} = \mathit{true}$ otherwise.

Suppose that $S_{i_1}, \ldots, S_{i_p}$ is a subcollection of $S_1, \ldots, S_n$ that covers T. Then we claim that the hypothesis $A_{i_1} = \mathit{true}$ *and* ... *and* $A_{i_p} = \mathit{true}$ is consistent with Q. To verify this, note that it clearly includes the positive example of Q and furthermore, because every x_i, $1 \leq i \leq k$ appears in some S_{i_l}, $1 \leq l \leq p$, every one of the negative examples has some attribute in $A_{i_1}, \ldots, A_{i_p}$ that is set to *false*, and thus is not included in this hypothesis.

On the other hand, if h is any pure conjunctive hypothesis that is consistent with Q, then h must have the form $A_{i_1} = \mathit{true}$ *and* .... *and* $A_{i_p} = \mathit{true}$ for some $\{i_1, \ldots, i_p\} \subseteq \{1, \ldots, n\}$, for otherwise it would not include the positive example of Q. Furthermore, each of the negative examples of Q must have the value *false* for some attribute in $A_{i_1}, \ldots, A_{i_p}$, otherwise it would be included in h. Because of the way the negative examples are defined, this implies that $S_{i_1}, \ldots, S_{i_p}$ cover T.

It follows that finding the minimum cover of T from the given collection of sets reduces to finding the pure conjunctive hypothesis that is consistent with Q and has the minimum number of atoms. Hence, since the minimim set cover problem is NP-hard, so is the problem of finding the smallest consistent pure conjunctive hypothesis. □

The above argument shows how the difficulty of finding the smallest consistent pure conjunctive hypothesis is related to the problem of finding the minimum cover of a set T among a collection of sets whose union is T. There is, however, an obvious heuristic for approximating the minimum cover of a set T: First choose a largest set in the collection. Then remove the elements of this set from T and choose another set that includes the maximum number of the remaining elements, continuing in this manner until T is exhausted. This is called the *greedy method.*

To apply this method to the problem of finding pure conjunctive concepts, we first make the following definition. Given an atom a involving an attribute A and a negative example, we say that *a eliminates* that negative example if it has a value for A that is not included in the set of values for A specified in a. For example, if a is the atom $2 \leq size \leq 5$, then a eliminates all negative examples that have sizes outside the range from 2 to 5. We now define the following algorithm.

Algorithm 5.2 (Greedy algorithm for learning pure conjunctive concepts).

Step 1. Find the minimal dominating atom for each attribute with respect to the given sample.

Step 2. Starting with the empty pure conjunctive hypothesis h, while there are negative examples in the sample do:

(a) Among all attributes, find the minimal dominating atom that eliminates the most negative examples and add it to h, breaking out of the loop if no minimal dominating atom eliminates any negative examples.

(b) Remove from the sample the negative examples that are eliminated.

Step 3. If there are no negative examples left return h, else report that the sample is not consistent with any pure conjunctive concept.

To see how this algorithm differs from Algorithm 4.1, consider again the same instance space and sample used in the previous section to illustrate Algorithm 4.1. The positive examples were

$$(square, 5.2, true)\ ,$$
$$(triangle, 3.4, true)\ ,$$
$$(square, 2.9, true)$$

and the negative examples were

$$(\textit{circle}, 4.3, \textit{true}),$$
$$(\textit{channel}, 5.1, \textit{true}),$$
$$(\textit{square}, 3.7, \textit{false}).$$

As in Algorithm 4.1, Step 1 of Algorithm 5.2 produces the set of minimal dominating atoms

$$\textit{shape} = \textit{regular\_polygon},$$
$$2.9 \leq \textit{size} \leq 5.2,$$
$$\textit{shade} = \textit{true}.$$

Initially the hypothesis h is empty. The atom $\textit{shape} = \textit{regular\_polygon}$ eliminates the most negative examples (two), so it is chosen first and conjoined to h. The examples that it eliminates are removed, leaving only one negative example ($\textit{square}, 3.7, \textit{false}$). The atom $\textit{shade} = \textit{true}$ eliminates this example, whereas the *size* atom does not, so it is now conjoined to h. All negative examples are now eliminated, so the hypothesis

$$\textit{shape} = \textit{regular\_polygon and shade} = \textit{true}$$

is returned. Because it omits the atom $2.9 \leq \textit{size} \leq 5.2$, this hypothesis is simpler than that produced by Algorithm 4.1.

It can readily be verified that, like Algorithm 4.1, Algorithm 5.2 uses the hypothesis space of all pure conjunctive concepts consistently. The proof is similar to that given in Lemma 4.2 and so is omitted. This means that the overall performance of Algorithm 5.2 is at least as good as that established for Algorithm 4.1 in Corollary 4.5.

However, Algorithm 5.2 has the additional property that, while it does not always find the simplest consistent conjunct, it does tend to find simpler conjuncts. This is guaranteed by the following bound on the approximation given by the greedy set cover heuristic.

Theorem 5.3 [18, 29]. *If the set T to be covered has m elements and s is the size of the minimum cover, then the greedy method is guaranteed to find a cover of size at most $s(\ln m + 1)$.*

From this theorem it follows that given m examples of an s-atom pure conjunctive concept, Algorithm 5.2 is guaranteed to find a consistent pure conjunctive hypothesis with at most $s(\ln m + 1)$ atoms. One way to look at this is as follows. Given a class of target concepts C to be learned, where in this case C is the class of all pure conjunctive concepts with at most s atoms, this algorithm learns concepts in C using a larger hypothesis space H, namely the class of all pure conjunctive concepts with at most $s(\ln m + 1)$ atoms, where m is the sample size. Algorithm 4.1 does this as well when applied to target concepts in C, except that in that case H is the class of all pure conjunctive concepts.

By using a larger hypothesis space than is strictly needed, it may be computationally easier to find a consistent hypothesis. This is certainly the case here. On the other hand, by using a larger hypothesis space, or more accurately, a hypothesis space with a larger growth function, more random examples will be required in general before we will have confidence that the hypothesis produced is a good approximation to the target concept. Thus it is important to strike a balance between the size or growth function of the hypothesis space and the computational difficulty of finding a consistent hypothesis in this space. Algorithm 5.2 does this in a particularly interesting way by, in effect, dynamically adjusting the size of its hypothesis space to the size of the sample and the complexity of the underlying target concept that generates the sample. This general technique leads to the following.

Definition 5.4. Let L be a learning algorithm, C be a class of target concepts and m be a sample size. By $H_C^L(m)$ we denote the set of all hypotheses produced by L from samples of size m of target concepts in C. We call $H_C^L(m)$ the *effective hypothesis space of* L for target concepts in C and sample size m.

The following corollary of Theorem 3.3 can now be used to obtain bounds on the learning performance of algorithms that dynamically adjust their hypothesis space according to sample size.

Theorem 5.5. *Let C be a class of target concepts and let L be a learning algorithm that always produces a consistent hypothesis (not necessarily in C) when given a sample of a target concept in C. Then given a sequence of $m \geq 1$ independent random examples (chosen according to any fixed probability distribution on the instance space) of any target concept $c \in C$, for any $0 < \varepsilon < 1$, the probability that L returns a hypothesis with error greater than ε is less than*

$$2\Pi_{H_C^L(m)}(2m)2^{-\varepsilon m/2} .$$

Proof. By Theorem 3.3, for any target concept c and distribution on the instance space, this is an upper bound on the probability that any hypothesis in $H_C^L(m)$ with error greater than ε is consistent with all m random examples of c. Since L always produces a consistent hypothesis in $H_C^L(m)$ for any sample of a target concept in C, when the target concept is in C this is therefore an upper bound on the probability that the hypothesis returned by L has error greater than ε. $\square$

In order to apply this result to obtain bounds on the sample complexity of Algorithm 5.2 we will use the following

Lemma 5.6. *If f is a real-valued function and there exist $a,b,d \geqslant 1$ such that $f(m) \leqslant a(bm)^{d \log m}$ for all $m \geqslant 2$, then there exists a constant c_1 such that*

$$f(m)2^{-\varepsilon m/2} \leqslant \delta$$

$$\textit{for all } 0 < \varepsilon,\delta < 1 \quad \textit{and}^7 \quad m \geqslant c_1(\log(a/\delta) + d(\log(bd/\varepsilon))^2)/\varepsilon\,.$$

Proof. This follows from [16, Lemma 1(iii)]. The calculations are outlined in [16, Appendix]. □

Corollary 5.7. *There are positive constants c_0 and c_1 such that for any instance space X defined on n attributes, each tree-structured or linear,*

$$c_0(\log(1/\delta) + s\log(n/s))/\varepsilon$$
$$\leqslant S_C^L(\varepsilon,\delta) \leqslant c_1(\log(1/\delta) + s(\log(sn/\varepsilon))^2)/\varepsilon$$

for all sufficiently small ε and δ, where L is Algorithm 5.2 and C is the class of pure conjunctive concepts on X with at most s atoms, $s \leqslant n$. Moreover, this lower bound holds for any learning algorithm L.

Proof. As in Corollary 4.8, the lower bound follows from Theorem 4.7, using the lower bound on VCdim(C) given in Theorem 3.6(ii). For the upper bound, note that from Theorem 5.3 it follows that $H_C^L(m)$ is contained in the class of pure conjunctive hypotheses with at most $s(\ln m + 1)$ atoms. Thus by Theorem 3.6(ii)

$$2\Pi_{H_C^L(m)}(2m) \leqslant 2(2\sqrt{n}m)^{2s(\ln m+1)} \leqslant 2(2\sqrt{n}m)^{4s\log m} \quad \text{for } m \geqslant 2\,.$$

Now let $a = 2$, $b = 2\sqrt{n}$ and $d = 4s$. Then by Lemma 5.6 there exists a constant c_1 such that

$$2\Pi_{H_C^L(m)}(2m)2^{-\varepsilon m/2} \leqslant \delta$$

$$\textit{for all } 0 < \varepsilon,\delta < 1 \quad \textit{and} \quad m \geqslant c_1(\log(1/\delta) + s(\log(sn/\varepsilon))^2)/\varepsilon\,.$$

Hence, by Theorem 5.5, for any distribution on the instance space, given a random sample of this size of any target concept in C, the probability that L produces a hypothesis with error greater than ε is at most δ. Thus, this is an upper bound on the sample complexity of L for targets concepts in C. □

[7] The $(\log(bd/\varepsilon))^2$ factor in this equation can be improved to $(\log(d/\varepsilon))^2 + \log b \log((d/\varepsilon)\log b)$, which replaces the $(\log b)^2$ term with a $\log b \log\log b$ term (see [16]).

Note that in spite of the fact that the greedy heuristic comes with only a fairly weak guarantee as to the simplicity of the hypothesis it produces, it still performs nearly as well as the algorithm that exhaustively searches for the simplest consistent conjunct (equation (2)), and, more importantly, comes within a poly-logarithmic factor of the optimal sample complexity. Thus it successfully trades off only a small increase in the number of examples needed for a very significant decrease in computational complexity. We can estimate the computation time required by Algorithm 5.2 as follows.

As in the previous section, for simplicity, we assume that for a linear attribute the time required to compare two values is constant and for a tree-structured attribute the time required to determine if one value is in the subtree below another value or to compute the least common ancestor of two values is constant. Thus the time required for finding the minimal dominating atom for a single attribute with respect to a sample of size m and determining how many negative examples it eliminates is $O(m)$.

Under these assumptions, a simple implementation of Algorithm 5.2 would take time $O(nm)$ for Step 1, $O(n)$ for each execution of Step 2(a) and $O(nz)$ for each execution of Step 2(b), assuming that in Step 2(b) the number of negative examples removed from the sample is z and we also update an array that maintains the number of negative examples eliminated by each minimal dominating atom in light of this new, smaller sample. (Initialization of this array can be done in Step 1 at no additional cost.) Since the total number of negative examples removed from the sample during the course of the algorithm is less than m, the total time spent in Step 2(b) is $O(nm)$. By the performance bound on the greedy method given Theorem 5.3 above, the total number of iterations of the loop of Step 2 is bounded by $O(s \log m)$, where s is the number of atoms in the target concept, so the total time spent in Step 2(a) is $O(ns \log m)$. Thus, the overall time is bounded by $O(n(m + s \log m))$. Often we will have $s \log m \leq m$, in which case this algorithm is optimal to within a constant factor.

6. Learning Pure Disjunctive, *k*-DNF and *k*-CNF Concepts

The complements of pure conjunctive concepts can be represented as pure disjunctive concepts. Hence this is the *dual form* of pure conjunctive concepts. A variant of Algorithm 5.2 can be used to learn pure disjunctive concepts. For a pure disjunction to be consistent with a sample, each atom must eliminate all negative examples and need only include some subset of positive examples, and all atoms together must include (cover) all positive examples. To achieve this, in place of minimal dominating atoms we use their dual counterparts, which we call *maximal subordinate atoms*. For each attribute A, these are the most general elementary atoms involving A that include at least one positive example and no negative examples. For tree-structured attributes, they are nodes closest to the root that define subtrees whose leaves contain only values from positive examples. For linear attributes, they are maximal intervals that contain only values from positive examples. Note that unlike minimal dominating atoms, each attribute can have more than one maximal subordinate atom.

The dual greedy method is to repeatedly choose the maximal subordinate atom that covers the most positive examples and add it to the hypothesis, removing any new positive examples that are covered, until either all positive examples are accounted for, or no maximal subordinate atom covers any of the remaining positive examples.

As in the previous section, this method produces a consistent pure disjunctive hypothesis if any exist, and this hypothesis has at most $s(\ln m + 1)$ atoms for any sample of size m of a pure disjunctive target concept with at most s atoms. The VC dimension and the growth function for the hypothesis space of pure disjunctive concepts with at most $s(\ln m + 1)$ atoms are bounded in the same way that the corresponding VC dimension and growth function for pure conjunctive concepts are bounded (Theorem 3.6(ii)). Hence the results given in Corollary 5.7 also hold when L is the learning algorithm defined by the dual greedy method and C is the target class of pure disjunctive concepts with at most s atoms.

The dual greedy method is a variant of the "star" methodology of Michalski [24]. However, in Michalski's method, you repeatedly pick a "seed" positive example at random and then add the (in this case unique) maximal subordinate atom that includes it to the hypothesis, removing any newly covered positive examples. The important difference here is that since we are using the greedy heuristic to select our maximal subordinate atoms rather than random draw, we are able to give quantitative performance bounds using the known bounds on the greedy method for set cover (Theorem 5.3). Michalski also suggests using the number of positive examples covered as a criterion for selecting between competing maximal subordinate atoms in more complicated learning domains where there can be more than one such atom for a given seed. However, this filtering comes later in his method, after the seed has already been randomly selected. This does not allow for the possibility that some seeds may be better than others for producing atoms that cover many positive examples.

The dual greedy method can be extended to learn k-DNF concepts for fixed k (see definition in Section 1). Apply the method as above, except at each step choose the k-atom pure conjunctive concept (term) that includes the most positive examples without including any negative examples. This choice can be made as follows.

For every attribute A and every pair of values v_1 and v_2 of A that occur among the examples of the sample, calculate either the least common ancestor v of v_1 and v_2 and form the atom $A = v$ (if A is a tree-structured attribute) or form the atom $v_1 \leq A \leq v_2$ (if A is linear and $v_1 \leq v_2$). This creates a pool of at most nm^2 atoms, where n is the number of attributes and m is the sample size.

For every conjunction of k atoms from this pool, determine the number of positive and negative examples it includes, and select the conjunction that includes the largest number of positive examples without including any negative examples. This can be done in time $O((nm^2)^k m) = O(n^k m^{2k+1})$.

The overall time analysis of the algorithm is now easy. By the basic performance bound on the greedy method (Theorem 5.3), given m examples of a k-DNF target concept with s terms, this method produces a consistent k-DNF concept with at most $s(\ln m + 1)$ terms. Hence, the main loop is executed at most $O(s \log m)$ times, giving an overall time bound of $O(sn^k m^{2k+1} \log m)$. (Since each iteration of the loop takes so long, we dispense with the more refined approach to the analysis taken in the previous section.)

In analogy with Corollary 5.7, we have the following bounds on the sample complexity of this algorithm.

Corollary 6.1. *There are positive constants c_0 and c_1 such that for any instance space X defined on n attributes, each tree-structured or linear,*

$$c_0(\log(1/\delta) + ks\log(n/ks^{1/k}))/\varepsilon$$
$$\leq S^L_C(\varepsilon,\delta) \leq c_1(\log(1/\delta) + ks(\log(ksn/\varepsilon))^2)/\varepsilon$$

for all sufficiently small ε and δ, where L is the above algorithm and C is the class of all k-DNF concepts on X with at most s terms, $k \leq n$ and $s \leq \binom{n}{k}$. Moreover, this lower bound holds for any learning algorithm L.

Proof. Similar to that of Corollary 5.7, but using Theorem 3.6(iii). □

Again, this shows that the sample complexity of the algorithm is within a poly-logarithmic factor of optimal. This improves on Valiant's result [40] for learning k-DNF by reducing the required sample size from $O(n^k)$ to a size logarithmic in n.

By duality, these results also extend to the class of k-CNF concepts. Theorem 3.6(iii) shows that the same bounds hold for the growth function and the VC dimension. Clearly the algorithm outlined above can be dualized again so that, as in Algorithm 5.2, in each step we choose the k-atom clause that includes all the positive examples and eliminates the most negative examples. If L is the resulting algorithm and C is the class of k-CNF concepts on n attributes with at most s clauses, then the same computational and sample complexity bounds derived above still hold.

This greedy method, like Valiant's method, is clearly computationally impractical for large k. Thus, in practice, the exhaustive search part of the algorithm should be replaced by a limited heuristic search (e.g. as in [24]). However, we have not found any heuristic techniques that lead to provably good learning performance for arbitrary distributions.

7. Internal Disjunctive Concepts

We now tackle the problem of learning internal disjunctive concepts. There are several ways to go about simplifying internal disjunctive hypotheses to improve the performance of a learning algorithm. One extreme is to try to get rid of as many compound atoms as possible, similarly to what we did with pure conjunctive hypotheses. The other is to try to reduce the number of internal disjunctions within one or more of the compound atoms of the hypothesis. A good compromise is to try to minimize the total number of atoms plus internal disjunctions in hypothesis, which we call the (syntactic) *size* of the hypothesis. For an internal disjunctive hypothesis h, the size of h is equal to the total number of occurrences of tree-structured attribute values and linear attribute value ranges in all compound atoms combined.

Let h be an internal disjunctive hypothesis that is consistent with a given sample. As with pure conjunctive hypotheses, each atom in h includes all positive examples and eliminates some set (possibly empty) of negative examples. A compound atom with this property will be called a *dominating compound atom*. We would like to eliminate all the negative examples using a conjunction of dominating compound atoms with the smallest total size. This leads to the following.

Minimum set cover problem with positive integer costs. Given a collection of sets with union T, where each set has associated with it a positive integer cost, find a subcollection whose union is T that has the minimum total cost.

Since it generalizes the minimum set cover problem, this problem is clearly NP-hard as well. However, approximate solutions can be found by a generalized greedy method. Let T' be a set of elements remaining to be covered. For each set in the collection, define the *gain/cost ratio* of this set as the number of elements of T' it covers divided by its cost. The generalized greedy method is to always choose the set with the highest gain/cost ratio and add it to the cover. As with the basic minimum set cover problem, it can be shown that if the original set T to be covered has m elements and s is the minimum cost of any cover, then the generalized greedy method is guaranteed to find a cover of cost at most $s(\ln m + 1)$ [8].

To apply this method to learning internal disjunctions, let the gain/cost ratio of a dominating compound atom be the number of negative examples it eliminates divided by its size.

Algorithm 7.1 (Greedy algorithm for learning internal disjunctive concepts).

Step 1. Starting with the empty internal disjunctive hypothesis h, while there are negative examples in the sample do:

(a) Among all attributes, find the dominating compound atom a with the highest gain/cost ratio and add it to h, breaking out of the loop if none have positive gains.

(b) Remove from the sample the negative examples a eliminates.

Step 2. If there are no negative examples left return h,[8] else report that the sample is not consistent with any internal disjunctive concept.

As an example, we trace the development of the hypothesis h in Algorithm 7.1, given the examples of Fig. 3(a). At the start of the algorithm h is empty. During the first iteration of the loop of Step 1 the dominating compound atom *shape = convex* is found to have the highest gain/cost ratio: it eliminates all the nonconvex negative examples in the bottom row of Fig. 3(a) and at a cost of 1, since only one value for *shape* is specified. Its gain/cost ratio is thus 4. This atom is thus added to h, giving

$$h = (\mathit{shape} = \mathit{convex})$$

and these four negative examples are removed from the sample. On the next iteration, the dominating compound atom $1.7 \leq \mathit{size} \leq 3.0$ is selected. It eliminates the large yellow square and the small red square for a gain of 2, at a cost of 1, because only one interval is specified. After this iteration

$$h = (\mathit{shape} = \mathit{convex})\ \mathit{and}\ (1.7 \leq \mathit{size} \leq 3.0).$$

On the next iteration, we find the atom *shape = regular_polygon or circle* has the highest gain/cost ratio ($\frac{1}{2}$), eliminating the ellipse at a cost of 2. Now

$$\begin{aligned} h = (\mathit{shape} = \mathit{convex})\ &\mathit{and}\ (1.7 \leq \mathit{size} \leq 3.0) \\ &\mathit{and}\ (\mathit{shape} = \mathit{regular\_polygon\ or\ circle}), \end{aligned}$$

which can be reduced to

$$h = (\mathit{shape} = \mathit{regular\_polygon\ or\ circle})\ \mathit{and}\ (1.7 \leq \mathit{size} \leq 3.0).$$

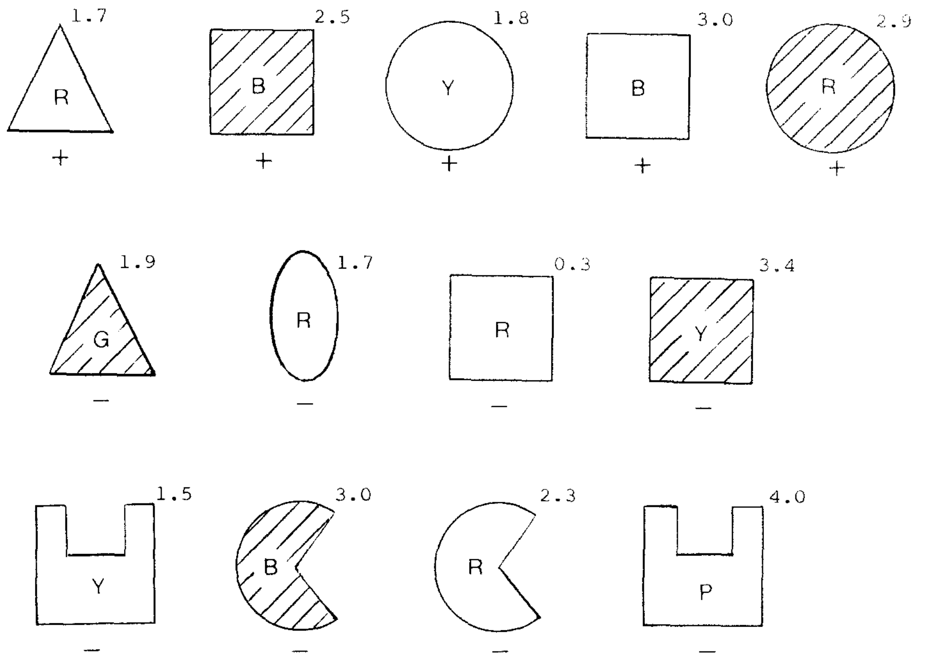
Finally, the last iteration eliminates the green triangle by adding the atom *color = red or yellow or blue*, giving the final hypothesis (in reduced form)

$$\begin{aligned} h = (&\mathit{shape} = \mathit{regular\_polygon\ or\ circle}) \\ &\mathit{and}\ (1.7 \leq \mathit{size} \leq 3.0)\ \mathit{and}\ (\mathit{color} = \mathit{red\ or\ yellow\ or\ blue}). \end{aligned}$$

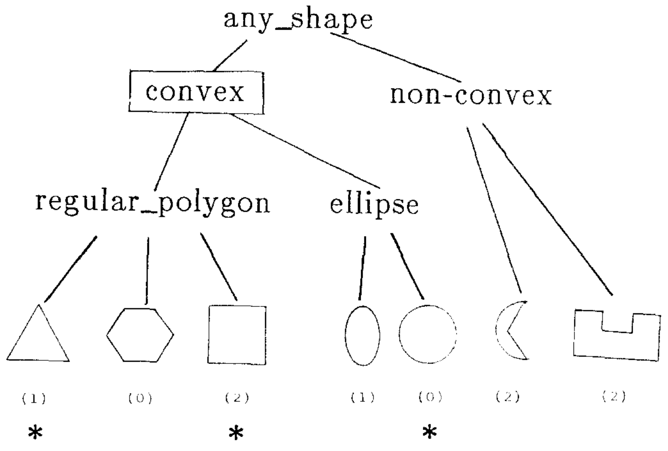
All negative examples have been eliminated, so this hypothesis is consistent with the sample, and is returned.

It is clear that to implement this algorithm, we need an efficient procedure to find a dominating compound atom with the highest gain/cost ratio for a given

[8] h may include several compound atoms for the same attribute. In practice these would be combined into one logically equivalent compound atom so that the final hypothesis is given in the simplest form.



(a)



(b)

Fig. 3. (a) Sample Q on an instance space defined by attributes: *shape*: as in Fig. 2; *color*: {*red* (R), *yellow* (Y), *blue* (B), *green* (G), *purple* (P), *orange* (O), *any_color*}; *size*: real-valued (values indicated next to example); *shade*: {*true* (|||), *false* ( ), *any_shade*}. (b) The minimal dominating atom for the attribute *shape* with respect to the sample Q of Fig. 3(a) is *shape* = *convex*. Values that appear in one or more positive example are marked with a star. The number in parentheses is the number of negative examples that the value appears in (used in Section 7).

attribute. Since there are in general exponentially many distinct dominating compound atoms with respect to the number of leaves of a tree-structured attribute or the number of values of a linear attribute, this cannot be done by an exhaustive search. However, there is a reasonably efficient dynamic programming procedure that does this for tree-structured attributes, and a simple iterative procedure for linear attributes. The reader that is not interested in the implementation details of these procedures can safely skip ahead to Corollary 7.2, where the learning performance of Algorithm 7.1 is evaluated.

The procedures we use to find a dominating compound atom with the highest gain/cost ratio for a given attribute actually produce what we call a *candidate list*, which is a list of dominating compound atoms with the highest gain, with one for each possible cost. We discuss the procedure for tree-structured attributes first.

Assume we are given a sample Q and a tree-structured attribute A. We first derive from A a tree T, called the *projection of Q onto A*, and two numbers, called the *base_gain* and *base_cost*. These objects are defined as follows. The leaves of T include only the leaves of A whose values occur among the positive examples of Q. The internal nodes of T are all the least common ancestors in A of sets of leaves of T. The descendant relationship among the nodes in T is the same as it was in A. Hence, the root of T is the least common ancestor in A of the set of all leaves of T. Each internal node of T is labeled with the name of the value it represents, taken from A, and two nonnegative integers called the *gain* and *cost*. The gain of an internal node σ is the number of additional negative examples eliminated when σ is *expanded* in a dominating compound atom, i.e. when the value represented by σ is replaced by the disjunction of the values represented by its immediate successors in T. Assume the immediate successors of σ in T are $\sigma_1, \dots, \sigma_k$. The gain of σ is calculated by determining the total number of negative examples with values associated with leaves that are in the subtree of σ in A, but not in any of the subtrees of $\sigma_1, \dots, \sigma_k$ in A. The cost of σ is the increase in the size of a dominating compound atom containing σ when σ is expanded. The cost is simply the number of immediate successors of σ in T minus one. *base_gain*(T) is the number of negative examples eliminated by the dominating compound atom $A = v$, where v is the value represented by the root of T. *base_cost*(T) is the cost of this atom, i.e. one. The projection of the sample Q given in Fig. 3(a) onto the attribute *shape* given in Fig. 3(b) is illustrated in Fig. 4.

By a *predecessor-closed subtree* of a tree T, we mean a subtree T' such that whenever a node σ of T is in T', then all predecessors of σ in T are also in T'. With each predecessor-closed subtree T' of the internal nodes of T, there is associated a *cut* of T, denoted *cut*(T'), defined as the set of all immediate successors of the leaves of T'. When T' is empty, *cut*(T') is the root of T. These definitions are illustrated in Fig. 5.

It is easily verified that the problem of finding a dominating compound atom

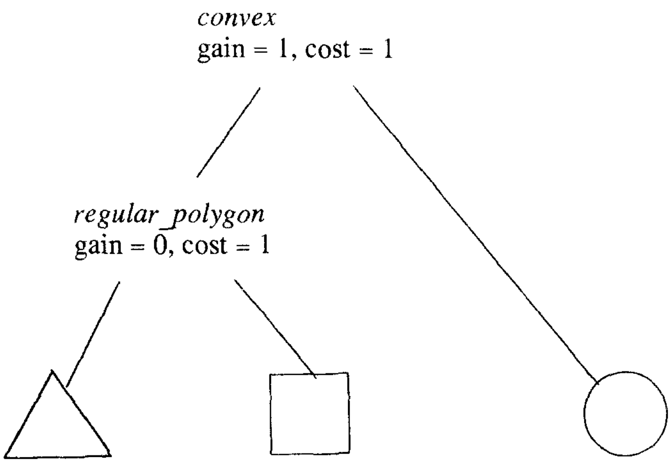


Fig. 4. Projection of Q onto *shape*; base_gain = 4, base_cost = 1.

for attribute A and sample Q with the highest gain/cost ratio can be reduced to finding $cut(T')$ in the projection T of Q onto A, where T' is the predecessor-closed subtree of the internal nodes of T that maximizes

$$\frac{base\_gain(T) + \sum_{\sigma \in T'} gain(\sigma)}{base\_cost(T) + \sum_{\sigma \in T'} cost(\sigma)}.$$

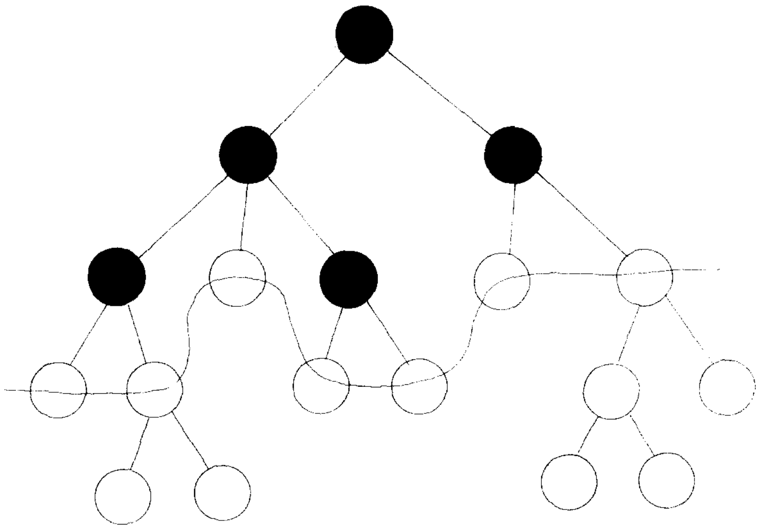


Fig. 5. A predecessor-closed subtree (● nodes) and its associated cut (⊖ nodes).

By adding a "dummy root" to T that has gain *base_gain*(T) and cost *base_cost*(T), and deleting the leaves of T (see Fig. 6), the latter problem reduces to the following:

Investment problem. Given a set of investments I, each of which has a nonnegative gain and a nonnegative integer cost, and a rooted tree T with node set I specifying which investments in I must be made prior to other investments (investment σ must be made before investment β if σ is an ancestor of β in the tree), find a feasible investment scheme with the highest gain/cost ratio, i.e. a nonempty predecessor-closed subtree T' of T that maximizes

$$\sum_{\sigma \in T'} gain(\sigma) \Big/ \sum_{\sigma \in T'} cost(\sigma) .$$

This investment problem is a variant of the similar investment problem solved in [19] by dynamic programming. There we are given a bound B on the maximum total cost of the investments we can make and seek to maximize our gain subject to this constraint. The dynamic programming technique given in [19] solves this problem by (essentially) calculating for each possible total cost the predecessor-closed subtree with the maximal total gain that has at most that cost. Not only does this solve our investment problem as well, but, under the above reduction, the cuts for these subtrees form the candidate list used for selecting the dominating compound atom with the highest gain/cost ratio. The combined time required for these calculations is $O(tq)$, where t is the number nodes in the tree and q is the number of distinct possible total costs.[9] When we

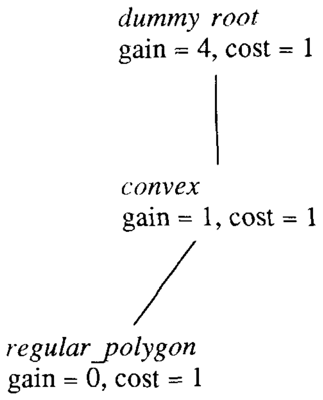


Fig. 6. Investment problem derived from Fig. 4.

[9]Because the algorithm runs in time proportional to the sum of the costs of the nodes, rather than the total number of bits required to represent these costs, it is only a pseudo-polynomial time algorithm [13]. In this case this is likely to be the best one can hope for, since the investment problem with bound B is NP-hard [19]. We do not know if the investment problem we have given above is also NP-hard.

are producing a candidate list from a projection, the size t of the tree is proportional to the number of distinct values for the attribute A that appear in positive examples in the sample, which is bounded by the sample size m. Since each possible total cost corresponds to the size of some subtree of this tree, the number of distinct possible total costs is also bounded by m, giving an $O(m^2)$ procedure. Of course this is a considerable overestimate if the trees for the attributes are small and the sample size is large.

We can calculate the candidate list for a linear attribute by a much simpler procedure. To do this, we partition the sequence of ordered values of the attribute using the maximal subordinate atoms, i.e. the maximal intervals that contain at least one positive example and no negative examples. The intervals between two consecutive intervals for maximal subordinate atoms will be called *gaps*. We rank the gaps in increasing order according to the number of negative examples that have their value in the gap. By selecting i gaps of highest rank for any $i \geq 0$, we can find a dominating compound atom of cost $i + 1$ with the maximum gain as follows.

First remove all other gaps by (temporarily) throwing away all negative examples with values that lie in these other gaps. Then form the dominating compound atom consisting of the disjunction of all the maximal subordinate atoms for the resulting sample. Since there will be only i gaps between consecutive intervals of maximal subordinate atoms, there will be only $i + 1$ atoms, hence the resulting compound atom will have cost $i + 1$. By construction, it will cover all positive examples (hence be dominating) and have maximal gain among compound atoms of the same size, since it eliminates all the negative examples with values in the i highest rank gaps, plus all the negative examples that do not lie in a gap because they have values that are either smaller than the smallest positive value or larger than the largest positive value. To make this procedure more compatible with the one for the tree-structured attributes, we then shrink each of the intervals in the compound atom as far as possible without uncovering any positive examples. This makes each interval the most specific that covers its positive examples, just as each atom formed by computing the least common ancestor in a tree-structured attribute of a set of positive examples is the most specific that covers these examples. Finally, to form the entire candidate list we do this for each cost i from 0 to the total number of gaps, hence this procedure, like the one for tree-structured attributes, also takes time quadratic in the number of distinct values of the attribute that appear in positive examples, and thus is $O(m^2)$.

The overall analysis of Algorithm 7.1 can now be given. Let s denote the size of the internal disjunctive target concept. The bound on the generalized greedy method guarantees that the loop in Step 1 of Algorithm 7.1 will be executed at most $O(s \log m)$ times, where m is the sample size. The cost of each iteration is dominated by the time it takes to produce the candidate lists for each attribute, which is $O(nm^2)$, giving an overall time bound of $O(snm^2 \log m)$. Again, this is a considerable overestimate if the number of distinct values for any attribute that appear in positive examples is small.

Finally, we can also give fairly tight bounds on the learning performance of Algorithm 7.1.

Corollary 7.2. *There are positive constants c_0 and c_1 such that for any instance space X defined on n attributes, each tree-structured or linear,*

$$c_0(\log(1/\delta) + s\log(n/s))/\varepsilon \le S_C^L(\varepsilon,\delta) \le c_1(\log(1/\delta) + s(\log(sn/\varepsilon))^2)/\varepsilon$$

for all sufficiently small ε and δ, where L is Algorithm 7.1 *and C is the class of internal disjunctive concepts on X with size at most s, $s \le n$. Moreover, this lower bound holds for any learning algorithm L.*

Proof. Similar to that of Corollary 5.7. □

This shows that the sample complexity of Algorithm 7.1 is also within a poly-logarithmic factor of optimal.

8. Conclusion

This work provides one step toward putting the empirical investigations in concept learning since Winston [45] on a solid theoretical foundation. We have taken the popular theme of inductive bias and formalized it quantitatively, relating this measure directly to learning performance. In so doing we have shown that simple, near-optimal learning algorithms exist for the well-studied classes of conjunctive, internal disjunctive, k-DNF and k-CNF concepts. With the exception of the algorithms for k-DNF and k-CNF concepts when k is large, these learning algorithms are also computationally efficient. Furthermore, the method we have developed is also quite general. In principle, it can be applied to any algorithm that learns single concepts from examples. It is required only that the algorithm produce consistent hypotheses, and that the hypothesis space used have a polynomially bounded growth function.

Nevertheless, the theoretical framework we have used here for analyzing concept learning algorithms is still inadequate on several accounts. First, we have made no mention of the possibility of misclassifications in the training sample. It is not clear how our algorithms could be modified to tolerate such misclassifications. Since all our general theorems demand that the hypothesis be consistent with the training sample, they would also need to be modified to deal with learning situations that involve misclassifications of the training examples. Clearly any practical theory of concept learning must deal with this possibility.

There are a number of approaches here. The methodology that Vapnik and others have used for pattern recognition starts from the general assumption that there is a fixed probability distribution on the set of all possible examples (i.e. instances and their labels). Hence, each instance may at times be classified as either "+" or "−", and the probability of a "+" classification may vary arbitrarily from instance to instance. Special cases of this general framework can be used to model many common types of "noisy" training data and/or "fuzzy" target concepts, depending on your point of view. A generalization of Theorem 3.3 given in [5, Appendix] (derived from [42]) is sometimes useful in such cases. In [2] a noisy training data viewpoint is adopted in their development and analysis of a noise resistant learning algorithm for k-CNF concepts in Boolean domains. Valiant has introduced a completely different model in which an adversary to the learning algorithm is allowed to maliciously modify the training examples [40]. This model is further developed in [20]. It is still not clear which, if any, of these "noise" models will be most appropriate for AI concept learning work.

Second, the methodology we have proposed may not be the most appropriate one for incremental learning algorithms, i.e. algorithms that maintain a working hypothesis and update this hypothesis as new examples are received. In many applications it is desirable to have a learning algorithm that works in this way. It does not appear that the algorithms based on the greedy heuristic that we have given can be used in such applications, short of storing all examples and recomputing the updated hypothesis from scratch when each new example is received. This problem has been addressed by the recent results of Littlestone [23]. For Boolean domains, he develops extremely efficient incremental algorithms for pure conjunctive and k-DNF concepts with performance very similar to those given here. These algorithms are based on a new variant of the perceptron learning algorithm, and are thus eminently suited for implementation in a "connectionist" architecture as well. However, they do not always maintain a consistent hypothesis, and much of their analysis appears to require mathematical techniques fundamentally different from those used here. The VC dimension is still used to provide lower bounds on the learning performance, however.

Third, the techniques here have been applied only to passive learning algorithms, i.e. algorithms that simply receive examples and form hypotheses. Angluin and others have demonstrated the power of learning algorithms that can also make *queries*, e.g. ask questions of the form "is x an instance of the target concept?," where x is an instance constructed by the learning algorithm [1, 36, 37] (see also [39]). Any comprehensive theoretical foundation for concept learning should also encompass such algorithms.

Fourth and finally, the methodology given here should be extended to richer types of instance spaces, to learning problems for multi-valued functions, to allow kinds of domain-specific background knowledge other than just orders and hierarchies on attribute values, and to learning problems that require simultaneously learning sets of related concepts. In [15] one extension to structured instance spaces (e.g. the blocks world [45]) is given. Background knowledge and multi-valued functions are considered in [28]. In [42] the basic methodology used here is extended to real-valued functions. Some richer types of background knowledge are considered in [25]. In [14] a few speculations on the issue of simultaneously learning sets of related concepts are given, based on ideas from [3, 7]. Much more work remains to be done in all these areas.

Apart from extending the analytical methodology presented here to overcome the above mentioned shortcomings, a number of other significant open problems remain within the present framework. We mention only two. First, how does the greedy heuristic we have used relate to Quinlan's information theoretic heuristic for learning decision trees [31]? Can the techniques given here be extended to exhibit efficient and provably effective learning algorithms for small decision trees in the Valiant framework? (See [34] and [11] for one approach here.) Second, is there an efficient and provably effective learning algorithm for simple DNF concepts, i.e. short DNF expressions with no fixed limit on the number of atoms per term? This problem was first posed by Valiant for Boolean domains, and still remains a central question today.

ACKNOWLEDGMENT

I would like to thank Larry Rendell for suggesting the relationship between the Vapnik–Chervonenkis dimension and Mitchell and Utgoff's notion of inductive bias, and Ryszard Michalski for suggesting I look at the problem of learning internal disjunctive concepts. I also thank Les Valiant, Leonard Pitt, Manfred Warmuth, Nick Littlestone, Phil Laird, Ivan Bratko and Stephan Muggleton and Andrzej Ehrenfeucht for helpful discussions of these ideas. I thank Ranan Banerji and Anselm Blumer for pointing out errors in an earlier version of this manuscript.

REFERENCES

1. Angluin, A., Queries and concept learning, *Machine Learning* **2** (4) (1988) 319–342.
2. Angluin, D., and Laird, P.D., Learning from noisy examples, *Machine Learning* **2** (4) (1988) 343–370.
3. Banerji, R., The logic of learning: a basis for pattern recognition and improvement of performance, *Adv. Comput.* **24** (1985) 177–216.
4. Blumer, A., Ehrenfeucht, A., Haussler, D. and Warmuth, M., Occam's razor, *Inf. Proc. Lett.* **24** (1987) 377–380.
5. Blumer, A., Ehrenfeucht, A., Haussler, D. and Warmuth, M., Learnability and the Vapnik–Chervonenkis dimension, *J. ACM*, to appear.
6. Bruner, J.S., Goodnow, J. and Austin, G.A., *A Study in Thinking* (Wiley, New York, 1956).
7. Bundy, A., Silver, B. and Plummer, D., An analytical comparison of some rule-learning programs, *Artificial Intelligence* **27** (1985) 137–181.
8. Chvatal, V., A greedy heuristic for the set covering problem, *Math. Oper. Res.* **4** (3) (1979) 233–235.
9. Cover, T., Geometrical and statistical properties of systems of linear inequalities with applications to pattern recognition, *IEEE Trans. Elect. Comput.* **14** (1965) 326–334.
10. Dietterich, T.G. and Michalski, R.S., A comparative review of selected methods for learning

from examples, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Tioga, Palo Alto, CA, 1983) 41–81.

11. Ehrenfeucht, A. and Haussler, D., Learning decision trees from random examples, Tech. Rept. UCSC-CRL-87-15, University of California, Santa Cruz, CA (1987).
12. Ehrenfeucht, A., Haussler, D., Kearns, M. and Valiant, L.G., A general lower bound on the number of examples needed for learning, *Inf. Comput.*, to appear.
13. Garey, M. and Johnson, D., *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, CA, 1979).
14. Haussler, D., Quantifying the inductive bias in concept learning, Tech. Rept. UCSC-CRL-86-25, University of California, Santa Cruz, CA (1986).
15. Haussler, D., Learning conjunctive concepts in structural domains, *Machine Learning*, to appear.
16. Haussler, D., Applying Valiant's learning framework to AI concept learning problems, Tech. Rept. UCSC-CRL-87-11, University of California, Santa Cruz, CA (1987); also in: R.S. Michalski and Kodratoff (Eds.), *Machine Learning* **III** (Morgan Kaufmann, Los Altos, CA 1987).
17. Haussler, D. and Welzl E., Epsilon nets and simplex range queries, *Discrete Comput. Geometry* **2** (1987) 127–151.
18. Johnson, D.S., Approximation algorithms for combinatorial problems, *J. Comput. Syst. Sci.* **9** (1974) 256–278.
19. Johnson, D.S. and Niemi, K.A., On knapsacks, partitions and a new dynamic programming technique for trees, *Math. Oper. Res.* **8** (1) (1983) 1–14.
20. Kearns, M. and Li, M., Learning in the presence of malicious errors, in: *Proceedings 20th ACM Symposium on Theory of Computing*, Chicago, IL (1988) 267–280.
21. Kearns, M., Li, M., Pitt, L. and Valiant, L., Recent results on Boolean concept learning, in: *Proceedings 4th International Workshop on Machine Learning*, Irvine, CA (1987) 337–352.
22. Laird, P.D., Inductive inference by refinement, Tech. Rept. YALEU/DCS/TR-376, Yale University, New Haven, CT (1986).
23. Littlestone, N., Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm, *Machine Learning* **2** (4) (1988) 245–318.
24. Michalski, R.S., A theory and methodology of inductive learning, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Tioga, Palo Alto, CA, 1983) 83–134.
25. Milosavljevic, A., Learning in the presence of background knowledge, Tech. Rept. UCSC-CRL-87-27, University of California, Santa Cruz, CA (1987).
26. Mitchell, T.M., The need for biases in learning generalizations, Tech. Rept. CBM-TR-117, Department of Computer Science, Rutgers University, New Brunswick, NJ (1980).
27. Mitchell, T.M., Generalization as search, *Artificial Intelligence* **18** (1982) 203–226.
28. Natarajan, B.K. and Tadepalli, P., Two new frameworks for learning, in: *Proceedings 5th International Workshop on Machine Learning*, Ann Arbor, MI (1988).
29. Nigmatullin, R.G., The fastest descent method for covering problems (in Russian), in: *Proceedings Symposium on Questions of Precision and Efficiency of Computer Algorithms* **5**, Kiev, U.S.S.R. (1969) 116–126.
30. Pearl, J., On the connection between the complexity and credibility of inferred models, *Int. J. General Syst.* **4** (1978) 255–64.
31. Quinlan, J.R., Induction of decision trees, *Machine Learning* **1** (1) (1986) 81–106.
32. Rendell, L., A general framework for induction and a study of selective induction, *Machine Learning* **1** (2) (1986) 177–226.
33. Rissanen, J., Modeling by shortest data description, *Automatica* **14** (1978), 465–471.
34. Rivest, R., Learning decision-lists, *Machine Learning* **2** (3) (1987) 229–246.
35. Schlimmer, J.C., Incremental adjustment of representations for learning, in: *Proceedings 4th International Workshop on Machine Learning*, Irvine, CA (1987) 79–90.

36. Sammut, C. and Banerji, R., Learning concepts by asking questions, in: R. Michalski, J. Carbonell and T. Mitchell (Eds.), *Machine Learning* **II** (Morgan Kaufmann, Los Altos, CA, 1986).
37. Subramanian, D. and Feigenbaum, J., Factorization in experiment generation, in: *Proceedings AAAI-86*, Philadelphia, PA (1986) 518–522.
38. Utgoff, P., Shift of bias for inductive concept learning, in: R. Michalski, J. Carbonell and T. Mitchell (Eds.), *Machine Learning* **II** (Morgan Kaufmann, Los Altos, CA, 1986).
39. Valiant, L.G., A theory of the learnable, *Commun. ACM*, **27** (11) (1984) 1134–1142.
40. Valiant, L.G., Learning disjunctions of conjunctions, in: *Proceedings IJCAI-85*, Los Angeles, CA (1985) 560–566.
41. Pitt, L. and Valiant, L.G., Computational limitations on learning from examples, Tech. Rept. TR-05-86, Aiken Computing Lab., Harvard University, Cambridge, MA (1986); also: *J. ACM*, to appear.
42. Vapnik, V.N., *Estimation of Dependences Based on Empirical Data* (Springer, New York, 1982).
43. Vapnik, V.N. and Chervonenkis, A.Ya., On the uniform convergence of relative frequencies of events to their probabilities, *Theor. Probab. Appl.* **16** (2) (1971) 264–280.
44. Wenocur, R.S. and Dudley, R.M., Some special Vapnik–Chervonenkis classes, in: *Discrete Math.* **33** (1981) 313–318.
45. Winston, P., Learning structural descriptions from examples, in: P.H. Winston (Ed.), *The Psychology of Computer Vision* (McGraw-Hill, New York, 1975).

Received November 1986; revised version received February 1988