On Learning More Concepts

Hussein Almuallim
Thomas G. Dietterich
Department of Computer Science
Oregon State University
Corvallis, OR 97331
almualh@cs.orst.edu
tgd@cs.orst.edu

Abstract

The *coverage* of a learning algorithm is the number of concepts that can be learned by that algorithm from samples of a given size. This paper asks whether good learning algorithms can be designed by maximizing their coverage. The paper extends a previous upper bound on the coverage of any Boolean concept learning algorithm and describes two algorithms—Multi-Balls and Large-Ball—whose coverage approaches this upper bound. Experimental measurement of the coverage of the ID3 and FRINGE algorithms shows that their coverage is far below this bound. Further analysis of Large-Ball shows that although it learns many concepts, these do not seem to be very interesting concepts. Hence, coverage maximization alone does not appear to yield practically-useful learning algorithms. The paper concludes with a definition of coverage *within a bias*, which suggests a way that coverage maximization could be applied to strengthen weak preference biases.

1 INTRODUCTION

Research in computational learning theory (e.g., [Valiant 84], [Natarajan 87], [COLT 88]–[COLT 91]) has provided many insights into the capabilities and limitations of inductive learning from examples. However, an important shortcoming of most work in this area is that it focuses on learning concepts drawn from *prespecified classes* of concepts (e.g., linearly separable functions, k-DNF formulae). This style of research begins by choosing a restricted class of concepts and then finding a polynomial bound—called the *sample complexity*—such that if a sample of size larger than the sample complexity is available, any concept from the concept class that is consistent with the sample will be approximately correct with high probability. Work of the above type usually leads to a learning algorithm that is *specialized* in learning the prescribed class of concepts, and an upper bound on the number of training examples required by the algorithm to guarantee successful learning.

For real-world applications, such findings can be viewed as follows: A learning algorithm L designed to learn a class of concepts $\mathcal{C}$ is guaranteed to succeed[1] in application domains in which the target concept belongs to $\mathcal{C}$ (i.e., the restrictions used to define $\mathcal{C}$ are satisfied by the target concept), provided that a sufficient number of training examples is given to the algorithm. Of course, no such guarantees are given if the target concept is not in $\mathcal{C}$.

This naturally means that one should seek algorithms that learn concept classes that are as large (i.e., less restricted) as possible. Rivest [Rivest 87], for instance, mentions this goal most explicitly by saying:

> "One goal of research in machine learning is to identify the largest possible class of concepts that are learnable from examples."

This goal is also declared (although less explicitly) in many papers in the related literature (e.g., [Valiant 84], [Natarajan 87], [COLT 88]–[COLT 91]).

Nevertheless, it is a well-known fact that learning larger classes of concepts necessarily requires a larger number of training examples [Blumer et.al. 87]. Such trade-off between the size of the class of concepts being learned and the required number of training examples dictates how far one can go in attempting to learn larger and larger classes of concepts.

Traditionally, this issue has been addressed by identifying new classes of concepts that are as large as possible but still require a training sample of size bounded by some polynomial. Such an approach, however, does not enjoy a great practical merit. In fact, the idea of learning prescribed classes of concepts in general suf-

[1] The guarantees are on being approximately correct with high confidence.

fers two important problems:

- Training examples are usually hard to obtain. In a typical inductive learning task, one has only a limited number of training examples, much less than the polynomial bounds provided by learning theory.
- The concept class is usually unknown. In most application settings, there is often considerable flexibility (and concomitant lack of prior knowledge) concerning the choice of which concept class to explore. In fact, many of the concept classes studied in computational learning theory have never been supported by any practical justification.

Due to these difficulties, the learning algorithms and sample complexity bounds developed in computational learning theory have rarely been of practical value.

Recently, an alternative theoretical framework was introduced [Dietterich 89]. Instead of fixing a class of concepts and then deriving the sample complexity, this framework turns the problem around by asking: Given a *fixed number* of training examples, what is the largest collection of concepts that some algorithm can learn? The intuition behind this framework is that, in the absence of additional information, one should prefer the learning algorithm that has the highest chance of learning the unknown concept—that is, the algorithm that can learn the largest number of concepts. In short, this framework could provide an approach to discovering an "optimal" bias for inductive learning in the absence of prior knowledge.

The goal of this paper is to explore this approach. We define the *coverage* of a learning algorithm to be the number of concepts learnable by the algorithm from a given sample size (and other relevant parameters). There are three questions raised by this approach:

1. For given sample size m, accuracy parameter ϵ and confidence parameter δ, what is the largest possible coverage that any algorithm can achieve?
2. Can we design a learning algorithm that attains this optimal coverage?
3. What is the coverage of existing learning algorithms?

This paper contributes to answering each of these questions. First, we generalize the upper bound on coverage given in [Dietterich 89]. Next, we present two learning algorithms and determine their coverage analytically. The coverage of the first algorithm, Multi-Balls, is shown to be quite close to the upper bound. The coverage of the second algorithm, Large-Ball, turns out to be even better than Multi-Balls in many situations. Third, we considerably improve upon Dietterich's limited experiments for estimating the coverage of existing learning algorithms. We find that the coverage of Large-Ball exceeds the coverage of ID3 [Quinlan 86] and FRINGE [Pagallo and Haussler 90] by more than an order of magnitude in most cases.

These results are very thought-provoking, because, upon careful analysis, it becomes clear that the Large-Ball algorithm is rather trivial and uninteresting. In the final part of the paper, we conclude that coverage analysis does not—by itself—provide a framework for deriving an optimal inductive bias. It does, however, provide a framework for designing optimal-coverage algorithms within a given bias.

2 DEFINITIONS AND NOTATION

We consider the space of Boolean concepts defined on n Boolean features. Let U_n be the set of all the 2^n truth assignments to the n features. A *concept* is an arbitrary set $c \subseteq U_n$. An *example* of a concept c is a pair $\langle X, c(X)\rangle$ where $c(X) = 1$ if $X \in c$ and 0 otherwise. The example is called *positive* in the first case, and *negative* in the second.

As in [Dietterich 89], we assume the uniform distribution over U_n. However, all our results can be easily extended to the distributions where the probability is 0 on a subset of U_n and uniform on the rest. This is done by substituting the number of instances in U_n having non-zero probability in place of every occurrence of 2^n in the results.

A *training sample* of a concept c is a collection of examples drawn randomly from U_n and labeled according to c. The number of examples in this collection is called the *sample size*, denoted by m. Except in our experimental work, we assume that examples in a sample are drawn independently (i.e., with replacement), and thus, a sample of size m does not necessarily contain m distinct examples. Note that this is different from [Dietterich 89], where sampling is done without replacement. Assuming that $m \ll 2^n$, however, this difference is not significant.

The *disagreement* between a training sample and a concept is the number of examples in the sample that are incorrectly classified by the concept.

The *distance* between two concepts c and h is the number of assignments $X \in U_n$ such that $c(X) \neq h(X)$. The *error* between c and h is the distance divided by 2^n, which is equivalent to the probability that a randomly chosen X will be classified differently by the two concepts. For any $0 < \epsilon < 1$, we say that h is ϵ-close to c if the error between the two concepts is at most ϵ. We let $Ball(c, \epsilon)$ denote the set of concepts that are ϵ-close to c. Note that for any concept$c' \in Ball(c, \epsilon)$, the distance between c' and c is at most $\epsilon 2^n$. Therefore, the number of concepts in $Ball(c, \epsilon)$ is given by

$$|Ball(c, \epsilon)| = \sum_{i=0}^{\lfloor \epsilon 2^n \rfloor} \binom{2^n}{i}$$

We call c and $\lfloor \epsilon 2^n \rfloor$ the *center* and *radius* of the ball, respectively.

A *learning algorithm* is a mapping from the space of samples to the space of concepts. The output of the algorithm is called an *hypothesis*. An hypothesis is *consistent* if it has no disagreement with the training sample.

We adopt PAC learning [Blumer et.al. 87] as the criterion for successful learning, but we restrict this to learning under the uniform distribution only. We say that an algorithm L *learns* a concept c for given m, ϵ and δ, if with probability at least $1-\delta$, L returns some hypothesis h that is ϵ-close to c when given a randomly drawn sample of c of size m, where the probability is computed over all the samples of c of size m. ϵ and δ are called the *accuracy* and *confidence* parameters, respectively.

In general, ϵ and δ are in the range $0 < \epsilon, \delta < 1$. In practice, however, only values that are close to 0 are interesting. For this reason, we will sometimes explicitly assume for instance that $0 < \epsilon < \frac{1}{4}$ and $0 < \delta < \frac{1}{2}$, with the understanding that these are reasonable assumptions in practice. Further, to simplify our results, we will only consider the values of ϵ such that $\epsilon 2^n$ is an integer. Clearly, this is not a serious assumption when n is sufficiently large.

For given n, m, ϵ and δ, the *coverage* of a learning algorithm is the number of concepts the algorithm learns with respect to these parameters.

3 UPPER BOUND ON COVERAGE

We begin by proving an upper bound on the best coverage that any algorithm can attain. An upper bound of this type has been proven for the case where the training sample is drawn randomly *without* replacement [Dietterich 89]. In the following, we generalize Dietterich's result and show that the same upper bound also holds for the case where sampling is done with replacement. In addition, we provide a closed-form expression for this bound.

Theorem 1 *Assuming that $m \leq (1-2\epsilon)2^n$, the coverage of any learning algorithm under the uniform distribution can not exceed*

$$\frac{2^m \sum_{i=0}^{\epsilon 2^n} \binom{2^n - m}{i}}{1-\delta}$$

concepts, for sample size m, accuracy parameter ϵ and confidence parameter δ.

Proof (sketch): The proof follows the one given in [Dietterich 89] except that it uses a probabilistic counting argument instead of a discrete counting argument in order to handle sampling with replacement. $\square$

It can be shown that for $0 < \epsilon < \frac{1}{4}$ and $m < \frac{1}{4}2^n$, the above quantity is further bounded above by

$$\frac{2^{(1-\epsilon \log_2 e)m+1} \sum_{i=0}^{\epsilon 2^n} \binom{2^n}{i}}{1-\delta}$$

which is again bounded above by

$$\frac{2^{(1-1.44\epsilon)m+1+H(\epsilon)2^n}}{1-\delta}$$

where $H(\epsilon) = \epsilon \log_2 \frac{1}{\epsilon} + (1-\epsilon)\log_2 \frac{1}{1-\epsilon}$.

This result shows that given a training sample of a reasonable size, any learning algorithm can learn only a small proportion of the concept space. As a numerical example, consider the case where $n = 20, m = 100,000$ and $\delta = \epsilon = 0.05$. In this case $H(\epsilon) \approx 0.286$. The above result states that no learning algorithm can learn more than

$$\frac{2^{92,788+0.286\times 2^{20}}}{0.95}$$

concepts. This is less than $\frac{1}{600,000}$ of the $2^{2^{20}}$ possible concepts definable over 20 features—a strikingly small fraction.

For the extreme case where $\delta = 0$, we can derive a much tighter bound:

Theorem 2 *If $\delta = 0$ and $\epsilon < \frac{1}{4}$, then the coverage of any learning algorithm is at most*

$$\sum_{i=0}^{\epsilon 2^n} \binom{2^n}{i}$$

concepts, for accuracy parameter ϵ and confidence parameter δ.

This suggests that Theorem 1 is not tight when δ is very small. It also implies that the degree of freedom provided by the confidence parameter, δ, in the PAC definition is very important. Any algorithm that does not exploit this freedom to output (with probability within δ) a totally incorrect hypothesis can have only very limited coverage.

4 THE MULTI-BALLS LEARNING ALGORITHM

Given these upper bounds on coverage, can we design algorithms that achieve these coverages?

Let c_1 and c_2 be two concepts with distance d, and suppose that we desire to construct a learning algorithm L that learns both c_1 and c_2. To do this, we must consider how L should treat every possible training sample consistent with c_1, c_2, or both.

Obviously, any sample that is consistent with only one of c_1 or c_2 can be mapped to some hypothesis that is

Algorithm Two-Balls (*Sample*)
1. If *disagreement*(*Sample*, c_1) $<$ *disagreement*(*Sample*, $\neg c_1$) then return c_1.
2. Else return $\neg c_1$. Break ties arbitrarily.

Figure 1: The Two-Balls Algorithm. c_1 is a Built-in Constant Concept.

within ϵ of the consistent concept. The key question is how to map a sample that is consistent with both c_1 and c_2. Now, if $\frac{d}{2^n} \leq 2\epsilon$, then there exists some concept h that is within ϵ of both c_1 and c_2. In this case, all we need to do is to map the sample to h. However, if $\frac{d}{2^n} > 2\epsilon$, then there exists no concept that is within ϵ of both c_1 and c_2, and therefore, we must map the sample either in favor of c_1 or in favor of c_2, but not both. In these cases, if the correct concept is c_2 and we map the sample in favor of c_1, we will commit a mistake, and we can only afford to do this with probability δ. The probability of getting a sample that is consistent with both c_1 and c_2 is $(\frac{2^n-d}{2^n})^m$, where m is the sample size. This quantity is decreasing as d increases, so if we choose d sufficiently large, we can keep the probability of a mistake below δ.

In short, if c_1 and c_2 are close together, then there is no problem, because we can choose an h ϵ-close to both. Conversely, if they are far apart, there is also no problem, because the probability of a mistake can be bounded by δ. This suggests that a good strategy for designing learning algorithms with high coverage is to choose a collection of concepts that is as large as possible, such that the distance between each pair of concepts in the collection is either

- sufficiently large to suppress the probability of getting a sample consistent with both concepts, or
- within $2\epsilon 2^n$, so that we can find concept(s) within ϵ of both concepts.

This means that the concepts to be learned must be clustered as one or more balls in the space of concepts. What we need to do in order to construct an appropriate algorithm is to keep the radius of each ball small enough, and at the same time, make the distance between the centers of the balls large enough.

As a trivial case, suppose that we want to learn the set of all concepts that are within ϵ of a fixed concept c—the set $Ball(c, \epsilon)$. This is achieved simply by returning c as the hypothesis regardless of the training sample. This leads to a coverage of $\sum_{i=0}^{\epsilon 2^n} \binom{2^n}{i}$.

A less trivial case is to learn 2 ϵ-balls of concepts. This is accomplished by the "Two-Balls" algorithm given in Figure 1. This algorithm returns, as the hypothesis, some fixed concept c_1 or its complement, whichever is *closer* to the training sample. For any concept c in $Ball(c_1, \epsilon)$, the probability of drawing an example of c that disagrees with c_1 (and thus, agrees with $\neg c_1$) is at most ϵ. The same argument applies to the concepts in $Ball(\neg c_1, \epsilon)$. Therefore, if $\epsilon \leq \delta$, then a sample of size 1 (that is, $m = 1$) is sufficient to learn these two balls. In general, we can show that a sample of size

$$m > \frac{12\epsilon}{(1-2\epsilon)^2} \ln \frac{1}{\delta}$$

is sufficient to learn the two balls as desired. Even when δ is as small as 0.001 (i.e. 99.9% confidence), for any ϵ in the range $0 < \epsilon \leq 0.1$, this evaluates to only 13 examples. This holds *independently* of n.

Since the sample size is usually much larger than this, a direct generalization of the above trivial cases is to attempt to learn as many balls of concepts as permitted by the sample size. The idea is to choose a collection of well-separated concepts (the centers of the balls) and attempt to learn all the concepts clustered around each of these centers. More specifically, we start by fixing two integers d and k, and then construct a set $\mathcal{H} = \{h_1, h_2, h_3, \cdots, h_k\}$ of k concepts such that the distance between each pair of concepts in $\mathcal{H}$ is at least d. Then given a training sample, the concept in $\mathcal{H}$ that has the minimum disagreement with the sample is returned as the hypothesis.

The goal of the algorithm is to learn all the concepts in

$$\bigcup_{h \in \mathcal{H}} Ball(h, \epsilon),$$

which will give a coverage of

$$k \sum_{i=0}^{\epsilon 2^n} \binom{2^n}{i}$$

provided that the intersection between the balls is empty. The question is, of course, how to determine the appropriate values of d and k such that this goal is accomplished. Particularly, we need to worry about the following:

1. As explained earlier, d must be large enough so that the interaction between concepts in different balls is kept within what is allowed by the confidence parameter δ.
2. The number of concepts that we can construct such that the minimum distance between each pair is d drops sharply as d increases. Therefore,

Algorithm Multi-Balls (*Sample*, ϵ, δ)

1. Find α in the range $2\epsilon < \alpha < \frac{1}{2}$ such that
   $1 - H(\alpha) = 2[1-\beta(\alpha)]^2\alpha^2\frac{m}{2^n}\log_2 e$
   where
   $H(\alpha) = \alpha\log_2\frac{1}{\alpha} + (1-\alpha)\log_2\frac{1}{1-\alpha}$, and
   $\beta(\alpha) = 1 + \frac{(\alpha-2\epsilon)^2}{8\alpha^3} - \sqrt{[1+\frac{(\alpha-2\epsilon)^2}{8\alpha^3}]^2 - 1}$.
2. Let $k = \lfloor 2^{2^n - H(\alpha)2^n} \times \frac{\delta}{2} \rfloor$.
3. Construct $\mathcal{H} = \{h_1, h_2, h_3, \cdots, h_k\}$ such that
   $\forall_{h_i,h_j\in\mathcal{H}}\ distance(h_i, h_j) \geq \lceil\alpha 2^n\rceil$.
4. Return some hypothesis in $\mathcal{H}$ that has minimal disagreement with *Sample* (break ties arbitrarily).

Figure 2: The Multi-Balls Algorithm.

making d too large causes k (and hence, the coverage of the algorithm) to be too small.

The following two lemmas show how to choose appropriate values for d and k.

Lemma 1 *For $0 < \epsilon < \frac{1}{4}$ and $2\epsilon < \alpha < \frac{1}{2}$, let $\mathcal{H} = \{h_1, h_2, \cdots, h_k\}$ be a set of concepts such that the distance between each pair $h_i, h_j \in \mathcal{H}$ is at least $d = \lceil\alpha 2^n\rceil$, and let $c \in Ball(h, \epsilon)$ for some $h \in \mathcal{H}$. Assume that L is a learning algorithm that on any sample S outputs some hypothesis $h_i \in \mathcal{H}$ that has minimal disagreement with S. Then, under the uniform distribution, the probability that a sample of c of size m is mapped by L to an hypothesis other than h is at most*

$$\min_{0<\beta<1} \quad k \cdot \{e^{-2(1-\beta)^2\alpha^2 m} + e^{-\beta\frac{(\alpha-2\epsilon)^2}{2\alpha}m}\} \ . \qquad (1)$$

Proof (sketch): Let $g \neq h$ be a *specific* concept in $\mathcal{H}$. We bound the probability that a sample of c is mapped to g (instead of h) by positioning c as far from h as possible (and yet still within $Ball(h, \epsilon)$). The probability can be calculated by nesting two binomial random variables, and, with some algebraic manipulation, this can be bounded by two applications of Hoeffding's lemma. We then multiply this bound by k to obtain the result. □

Note that d in this lemma is expressed as a fraction α of 2^n. This result says that if the conditions of the lemma are met, and if c is a concept in one of the k ϵ-balls, then the samples of c are usually mapped by L to the center of that ball (which is ϵ-close to c) except with a probability that is bounded by Equation (1). Thus, α and k must be chosen so that this probability is at most δ. It is important to note that this probability is diminishing in α and m and independent of n.

Lemma 2 *For any α, $0 < \alpha < \frac{1}{2}$, and any even positive integer l, we can construct at least $2^{l-\lceil H(\alpha)l\rceil}$ bit vectors of length l and pairwise distance at least $\lceil\alpha l\rceil$, where $H(\alpha) = \alpha\log_2\frac{1}{\alpha} + (1-\alpha)\log_2\frac{1}{1-\alpha}$.*

This lemma is derived from a result in the field of Error-Correcting Coding Theory known as the Gilbert-Varshamov bound. A proof of this result in addition to a method of constructing the l-bit vectors, can be found in [Peterson and Weldon 72].

Using Lemmas 1 and 2, one can search for the appropriate value for d that leads to learning k different ϵ-balls, for k as large as possible. Let's now compute a lower bound on the coverage that can be achieved by this approach.

Figure 2 shows the "Multi-Balls" algorithm in which we give a specific way of choosing the values of α (and hence, d) and k. To be able to give a lower bound on the coverage of this algorithm, we need the following definition.

Definition: For $0 \leq \theta \leq 1$ and $0 < \epsilon < \frac{1}{4}$, define $\rho(\theta, \epsilon)$ as

$$\rho(\theta, \epsilon) = \frac{1 - H(\hat{\alpha})}{\theta}$$

for $\hat{\alpha}$ being the solution of the equation[2]

$$1 - H(\alpha) = 2[1-\beta(\alpha)]^2\alpha^2\theta\log e$$

in the range $2\epsilon < \alpha < \frac{1}{2}$, where

$$\beta(\alpha) = 1 + \frac{(\alpha-2\epsilon)^2}{8\alpha^3} - \sqrt{[1+\frac{(\alpha-2\epsilon)^2}{8\alpha^3}]^2 - 1}$$

and

$$H(\alpha) = \alpha\log_2\frac{1}{\alpha} + (1-\alpha)\log_2\frac{1}{1-\alpha}.$$

□

[2]It can be shown that there always exists a solution for α as desired.

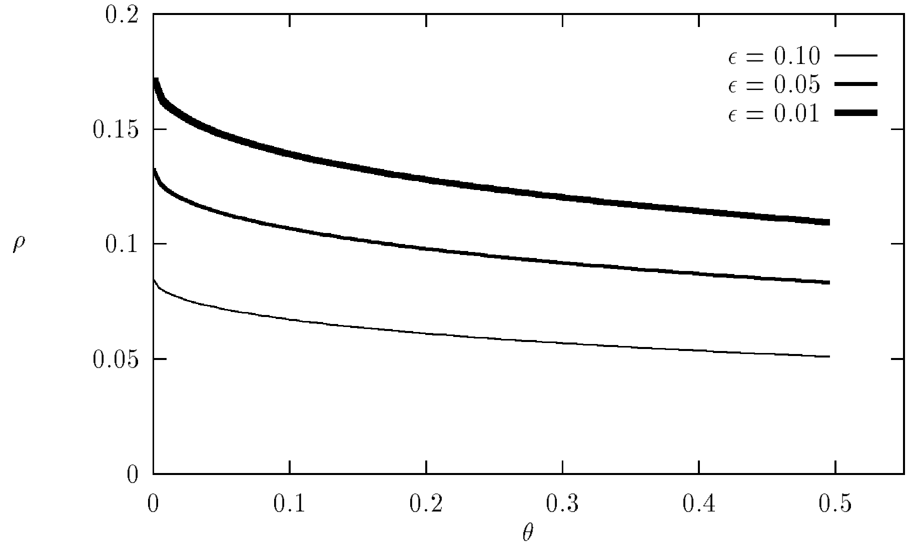

Figure 3: The Function $\rho(\theta, \epsilon)$.

Although $\rho(\theta, \epsilon)$ is not provided in closed form, it is easily computed for any given values of θ and ϵ using standard numerical methods. For illustration, Figure 3 plots this function over the range $0 \leq \theta \leq \frac{1}{2}$ for $\epsilon = 0.01, 0.05$ and 0.10.

Using the above definition of ρ, a lower bound on the coverage of Multi-Balls can be stated as follows:

Theorem 3 *For $0 < \epsilon < \frac{1}{4}$ and $0 < \delta < 1$, the coverage of the Multi-Balls algorithm under the uniform distribution is at least*

$$\left\lfloor \frac{\delta}{2} \; 2^{\rho(\frac{m}{2^n}, \epsilon)\, m} \right\rfloor \sum_{i=0}^{\epsilon 2^n} \binom{2^n}{i}$$

for sample size m, accuracy parameter ϵ and confidence parameter δ.

Proof (sketch): Substitute the quantities calculated in Steps 1 and 2 of the algorithm into Lemmas 1 and 2 to verify that k balls can be found and that the probability of mistake is bounded by δ. $\Box$

More specific bounds on the coverage of Multi-Balls can be obtained from Theorem 3 if upper bounds on $\frac{m}{2^n}$ and ϵ are assumed. For example, if we are interested only in the range $0 < \epsilon \leq 0.05$ and $0 \leq \frac{m}{2^n} \leq 0.25$ (which are reasonable assumptions in practice), then the coverage of Multi-Balls as given by Theorem 3 is at least

$$\left\lfloor \frac{\delta}{2} \; 2^{0.094\, m} \right\rfloor \sum_{i=0}^{\epsilon 2^n} \binom{2^n}{i}$$

where the constant 0.094 is just the value of $\rho(0.25, 0.05)$.

Note that the main difference between the lower bound of Equation(4) and the upper bound of Theorem 1 is the coefficient of m in the exponent of 2. Therefore, for any fixed δ, this lower bound indicates that to achieve a given coverage, Multi-Balls requires a sample size that is within a constant factor of that required by an optimal learning algorithm.

5 THE LARGE-BALL LEARNING ALGORITHM

So far we have been trying to maximize the coverage by learning as many balls of concepts as possible, while fixing the radius of each ball at $\epsilon 2^n$. An alternative approach to increase the coverage is to learn ball(s) of concepts with larger radius. Because of the extremely high dimensionality of the space of concepts, any small increment in a ball's radius results in a huge increase in the number of concepts contained in the ball.

It turns out that learning a single ball of radius larger than $\epsilon 2^n$ is a surprisingly easy task, as shown by the "Large-Ball" algorithm given in Figure 4. This algorithm works by modifying a *default* hypothesis c_1 so that the final hypothesis fully agrees with the training sample. For example, suppose c_1 is the *nil* concept. Then this algorithm classifies all examples as negative unless they appeared as positive examples in the training sample! The coverage of this algorithm is computed by the following theorem.

Theorem 4 *For sample size m, accuracy parameter ϵ and confidence parameter δ, the coverage of the Large-*

Algorithm: Large-Ball (*Sample*)
1. Let c_1 be a constant concept.
2. Define the concept s as: $s(X) = \begin{cases} 1 & \text{if } X \in \mathit{Sample} \\ 0 & \text{otherwise} \end{cases}$
3. Define the concept p as: $p(X) = \begin{cases} 1 & \text{if } X \in \mathit{Sample} \text{ and } X \text{ is a positive example} \\ 0 & \text{otherwise} \end{cases}$
4. Return $h = (\neg s \wedge c_1) \vee p$.

Figure 4: The Large-Ball Learning Algorithm.

Ball algorithm under the uniform distribution is

$$\sum_{i=0}^{\epsilon 2^n+\beta} \binom{2^n}{i}$$

where β is the largest integer such that

$$\sum_{k=0}^{\beta-1} \binom{\epsilon 2^n+\beta}{k} \left(\frac{2^n - \epsilon 2^n - \beta + k}{2^n}\right)^m \times$$

$$\{1 - \sum_{i=1}^{k} \binom{k}{i} \left(\frac{2^n - \epsilon 2^n - \beta + k - i}{2^n - \epsilon 2^n - \beta + k}\right)^m (-1)^{k+1}\} \leq \delta.$$

Proof (sketch): Without loss of generality, let c_1 be the *nil* concept. Let c be a concept having exactly $\epsilon 2^n + \beta$ positive examples. A sample of c is mapped by Large-Ball to an ϵ-far concept only if it contains less than β *distinct* positive examples. The theorem follows by showing that the probability of drawing such a sample is just the left-hand side of the inequality of the theorem. This is a variation of the Coupon-collecting problem (e.g., [Ross 88] p. 111). □

It can be shown that a sample of size

$$\frac{1}{\epsilon} \ln \frac{1}{\delta}$$

is sufficient to make β at least 1, and that in general, the value of β is at least

$$\frac{\epsilon m - \ln \frac{1}{\delta}}{n \ln 2 + \ln 2 - \ln \frac{1}{\epsilon}} \geq \frac{\epsilon m - \ln \frac{1}{\delta}}{n \ln 2}$$

provided that $\epsilon < \frac{1}{2}$. Since β grows linearly in m, the above theorem says that the coverage of Large-Ball grows quite rapidly as the sample size increases.

The coverage lower bound obtained for Large-Ball appears to overlap the bound for Multi-Balls, although Multi-Balls gives higher coverage for small values of ϵ (e.g. 0.01). In any case, the fact that a trivial algorithm like Large-Ball achieves such high coverage suggests that coverage analysis alone is not strong enough to derive good inductive biases. Before considering this point further, let us measure the coverage of some popular learning algorithms.

6 COVERAGE OF CURRENT LEARNING ALGORITHMS

A straightforward method to measure the coverage of a learning algorithm (as done in [Dietterich 89]) is to run it on every possible training sample. However, the great cost of this method limited Dietterich's experiments to concepts defined over only 3 features.

In a separate paper [Almuallim 91], we reported some techniques to reduce the computational costs involved in such experiments by resorting to statistical approximation and by exploiting the symmetry properties of learning algorithms with respect to permutation and negation of features. With these techniques, we can carry out coverage evaluation experiments on the space of concepts defined on up to 5 Boolean features. Due to space limitations, we only give the results of these experiments here. Please see [Almuallim 91] or [Almuallim 92] for more details.

In these experiments, three learning algorithms were considered: ID3 [Quinlan 86], FRINGE [Pagallo and Haussler 90] and MDT, which is an exhaustive algorithm that finds a decision tree with fewest nodes consistent with the training sample. The coverage of these algorithms was measured for $n = 5$, $\epsilon = \delta = 0.1$ and $m = 8, 10, 12, 14$, and 16. Sampling in these experiments was done without replacement. The results are summarized in Table 1.

In this table, BALLS denotes the algorithm that classifies all the examples that are not in the training sample as positive if the majority of the examples in the sample are positive, or as negative otherwise (breaking ties arbitrarily). For those examples included in the training sample, the algorithm gives the same class as given in the sample. The last row in the table gives the maximum coverage that can not be exceeded by any algorithm, using the upper bound of Theorem 1.

Note three points:

1. MDT does not give better coverage than the heuristic algorithms ID3 and FRINGE.
2. The coverage of ID3 and FRINGE is disappointingly smaller than that of Large-Ball and BALLS.

Table 1: Coverage Figures For Various Algorithms.

Algorithm	Sample size				
	8	10	12	14	16
ID3	12±0	332±0	396±0	1,756±0	4,954±640
FRINGE	12±0	332±0	396±0	1,756±0	5,284±970
MDT	12±0	12±0	116±40	496±0	3,694±0
Large-Ball	5,489	5,489	5,489	41,449	41,449
BALLS	10,978	10,978	10,978	82,898	82,898
Upper Bound	661,333	2,041,173	6,148,551	17,985,991	50,753,991

3. The coverage of all these algorithms is far below the upper bound of Theorem 1.

7 DISCUSSION

We began this paper by suggesting that an important design criterion for learning algorithms should be the coverage of the algorithm. We presented the Multi-Balls algorithm and showed that it can achieve optimal coverage with a sample size that is within a constant factor of optimal. However, we then showed that a fairly trivial algorithm, Large-Ball, can also achieve very large coverage—larger than Multi-Balls in cases where ϵ is reasonably big. Experimental tests confirm that Large-Ball and BALLS, a variation of Multi-Balls, have much better coverage than the popular ID3 algorithm and its relatives.

Why does Large-Ball strike us as trivial? Because it merely memorizes the training sample—it does not attempt to find any regularity in the data. Furthermore, the concepts it learns, while they are very numerous, are all located near a single concept. In short, the bias of Large-Ball is unlikely to be appropriate in real-world learning situations. This argument shows that coverage analysis alone is not sufficient to find a practically-useful inductive bias.

This suggests that we combine coverage analysis with other methods for choosing inductive bias. For example, in [Almuallim and Dietterich 91], we described learning situations in which the MIN-FEATURES bias—the bias that prefers consistent concepts definable over fewer features—is appropriate. However, the MIN-FEATURES bias does not uniquely define a learning algorithm, because, given a training sample, there are typically many consistent hypotheses that have the same, minimum, number of features. Hence, *within* the MIN-FEATURES bias, we could apply coverage analysis to design a learning algorithm that has the largest coverage among all algorithms that implement MIN-FEATURES.

In general, let $Pref(c_1, c_2)$ be a preference bias that prefers c_1 to c_2 in all cases where both concepts are consistent with the training sample. Let $Learns(L, c, m, \epsilon, \delta)$ be true if algorithm L can learn concept c from a sample of size m with accuracy and confidence parameters ϵ and δ. The *coverage within bias Pref* for L (with respect to m, ϵ, and δ), is the size of the set

$$C = \{c \mid Learns(L, c, m, \epsilon, \delta) \text{ and } \forall c' \; Pref(c', c) \Rightarrow Learns(L, c', m, \epsilon, \delta)\}$$

That is, a concept is "covered" only if all concepts preferred to it are also covered.

In conclusion, the results from this paper suggest that an important problem for future research is to design and analyze algorithms that have optimal coverage-within-bias for many of the popular biases. This will be particularly important for biases that are so weak that they do not have polynomial sample complexity.

Acknowledgements

Hussein Almuallim was supported by a scholarship from the University of Petroleum and Minerals, Saudi Arabia. The authors gratefully acknowledge the support of the NSF under grant number IRI-86-57316. Thanks to Rob Holte for useful discussions and for providing [Holte 91], and to Prasad Tadepalli for comments on an earlier draft of the paper.

References

[Almuallim and Dietterich 91] Almuallim, H. and Dietterich, T. G. Learning With Many Irrelevant Features. Proceedings of the 9th National Conference on Artificial Intelligence (AAAI-91), 547–552, 1991.

[Almuallim 91] Almuallim, H. Exploiting Symmetry Properties in the Evaluation of Inductive Learning Algorithms: An Empirical Domain-Independent Comparative Study. Technical Report, 1991-30-09, Dept. of Computer Science, Oregon State University, Corvallis, OR 97331-3202.

[Almuallim 92] Almuallim, H. Concept Coverage and Its Application to Two Learning Tasks. Ph.D. Thesis. Department of Computer Science, Oregon State University, Corvallis, Oregon. Forthcoming, 1992.

[Blumer et.al. 87] Blumer, A.; Ehrenfeucht, A.; Haussler, D.; and Warmuth, M. Learnability and the Vapnik-Chervonenkis Dimension, Technical Report UCSC-CRL-87-20, Department of Computer and Information Sciences, University of California, Santa Cruz, Nov. 1987. Also in *Journal of ACM,* 36(4):929–965, 1990.

[Chernoff 52] Chernoff, H. A Measure of Asymptotic Efficiency for Tests of Hypothesis Based on the Sums of Observations. Annals of Mathematical Statistics, 23, pp. 493–509, 1952.

[COLT 88] Proceedings of the 1988 Workshop on Computational Learning Theory. Haussler, D. and Pitt, L., Editors. Morgan Kaufmann Publishers, 1988.

[COLT 89] Proceedings of the Second Annual Workshop on Computational Learning Theory. Rivest, R., Haussler, D. and Warmuth, M.K., Editors. Morgan Kaufmann Publishers, 1989.

[COLT 90] Proceedings of the Third Annual Workshop on Computational Learning Theory. Fulk, M.A. and Case, J., Editors. Morgan Kaufmann Publishers, 1990.

[COLT 91] Proceedings of the Fourth Annual Workshop on Computational Learning Theory. Valiant, L.G. and Warmuth, M., Editors. Morgan Kaufmann Publishers, 1991.

[Dietterich 89] Dietterich, T. G. Limitations on inductive learning. In Proceedings of the Sixth International Conference on Machine Learning, 124–128. Ithaca, NY: Morgan Kaufmann, 1989.

[Hoeffding 63] Hoeffding, W. Probability Inequalities for Sums of Bounded Random Variables. *Journal of The American Statistical Association,* 58, 13-3-, 1963.

[Holte 91] Holte, R. C. Machine Learning as Error-Correction. Unpublished note, 1991.

[Natarajan 87] Natarajan, B.K. On learning Boolean Functions. In Proceedings of the 19th Annual ACM Symposium on Theory of Computing 296–304. New York, NY, 1987.

[Pagallo and Haussler 90] Pagallo, G.; and Haussler, D. Boolean feature discovery in empirical learning. *Machine Learning*, 5(1):71–100, 1990.

[Peterson and Weldon 72] W. W. Peterson and E. J. Weldon. Error Correcting Codes, The MIT Press. p.86, 1972.

[Quinlan 86] Quinlan, J. R. Induction of Decision Trees, *Machine Learning,* 1(1):81–106, 1986.

[Rivest 87] Rivest, R. Learning Decision Lists. *Machine Learning,* 2(3):229–246, 1987.

[Ross 88] Ross, S. A First Course in Probability. Macmillan Publishing Company, New York. 3rd edition, pp 111, 1988.

[Valiant 84] L. G. Valiant. A Theory of the Learnable. *Communications of ACM,* 27(11):1134-1142, 1984.