NOTICE WARNING CONCERNING COPYRIGHT RESTRICTIONS:
The copyright law of the United States (title 17, U.S. Code) governs the making of photocopies or other reproductions of copyrighted material. Any copying of this document without permission of its author may be prohibited by law.

Extending the Valiant Framework to Detect Incorrect Bias

Lonnie Chrisman

May 3, 1989
CMU-CS-89-137

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15217

Abstract

Since it is difficult to know the correct bias for an inductive learning problem *a priori*, the ability to detect a bad bias can be valuable. One method is to take advantage of an algorithm that makes strong performance guarantees when the bias *is* correct, then verify that the algorithm performs as promised. This paper develops this idea within the context of the Valiant framework.

In the basic Valiant framework, *if* the assumption that the target concept belongs to a given concept class holds *then* the output of a learning algorithm is $(1-\delta)$-reliable. After incorporating the notion of bias-evaluation, the assumption is removed such that the output is $(1-\delta)$-reliable regardless of whether or not the target concept belongs to the given concept class. It is shown how to convert an existing *pac*-learning algorithm into one with reliable bias-evaluation.

This research was sponsored by NASA under contract number NAGW-1175. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of NASA.

Extending the Valiant Framework to Detect Incorrect Bias

Lonnie Chrisman

1 Introduction

The most successful theoretical framework to date for learning from examples has been the Valiant model [Valiant 84] for what [Angluin 88] has termed *probably approximately correct* identification (or *pac*-identification). Its success stems primarily from the fact that several important concept classes (eg. k-DNF) have been shown to be *pac*-learnable. The Valiant model derives its power from the assumption that the concept being learned, the target concept, belongs to a given known restricted concept class (contemporarily referred to as the *inductive bias* [Mitchell 80]). However, as [Amsterdam 88a] points out, "Nature does not tell us that her regularities are describable by k-CNF." In certain cases, it may be possible to deduce an appropriate bias prior to learning (cf. [Russell & Grosof 87]) and in other cases the correct bias may be provided by a benevolent teacher. Unfortunately, when the possibility exists that a learner may pick an incorrect bias, the sources of power for the Valiant framework (specifically the guarantees of accuracy, reliability, and computational efficiency) disappear. *The Valiant model alone makes no performance guarantees when the learner's bias can be incorrect*[1].

As a very simple example, consider the trivial learning algorithm that is biased into believing that the target concept belongs to the set $\{TRUE, FALSE\}$. The algorithm examines a single classified training instance and returns its classification as the learned concept. If the bias is correct, the algorithm can guarantee that with a single training example it will return an exactly correct concept 100% of the time. However, if the possibility exists that the target concept does not belong to $\{TRUE, FALSE\}$, this algorithm does not make any significant guarantees about the accuracy or reliability of the resulting concept.

Since one dimension of success for a theoretical model is its utility in actual applications, an extension which would guarantee robust performance even in the presence of an incorrect bias would represent a significant contribution to the Valiant framework. This paper presents such an extension. The approach here gives a *pac*-learning algorithm the option of returning a reliably accurate concept (as in the unmodified Valiant framework) or alternatively it may report that the bias is bad. Whichever response the algorithm chooses, its output can be regarded as reliable (in a sense that is defined precisely in Section 3) whether or not its bias is correct. Since such an algorithm inherently possesses some type of ability to evaluate the correctness of its assumptions about the target concept, an algorithm with these qualities (and polynomial-time complexity) is termed a *bias-evaluating pac-learning algorithm.*

After presenting the basic Valiant model for *pac*-identification in Section 2, Section 3 will introduce the precise requirements for a bias-evaluating *pac*-learning algorithm. It is shown

[1] However, [Kearns *et.al* 87] and [Amsterdam 88a,b] introduce extensions to the Valiant framework which are noteworthy with respect to this problem. Also, *Hypothesis Filtering* [Etzioni 88] can be used to preserve accuracy and reliability. See Sections 3.3 and 4 of this paper for a discussion of related work.

UNIVERSITY LIBRARIES
CARNEGIE-MELLON UNIVERSITY
PITTSBURGH, PENNSYLVANIA 15213

in Section 3.1 how a bias evaluator consisting of a simple statistical sampling test, similar in nature to *Hypothesis Filtering* [Etzioni 88], can be appended to any *pac*-learning algorithm. The construction leads to the paper's main result that under one very reasonable assumption, a bias-evaluating *pac*-learning algorithm exists for any *pac*-learnable concept class. I conclude with a discussion of related work in Section 4.

2 The Valiant Framework

The Valiant framework was introduced in [Valiant 84] as a model for inductively learning concepts from specific examples. Rather than requiring a learner to always find an exact concept, the model requires only that the learner produces with high probability a sufficiently close approximation to the actual target concept. A learning algorithm is also required to be computationally efficient.

We assume there is a space X of possible instances where each instance is represented by n attribute values. A learning algorithm may request from an oracle $EX()$ a classified training example selected randomly from X according to some fixed but unknown probability distribution D[2]. It is assumed that the distribution remains fixed even after the learning session has been completed. A concept is any subset of X; in particular, the target concept L_* is the concept the system is attempting to learn. When the randomly selected training example belongs to L_*, $EX()$ classifies it as positive; otherwise, it is classified as negative. The basic Valiant model makes no assumptions about the form of the distribution D.

The task of the *pac*-learner is to produce a concept L_h which differs from the target concept L_* by no more than some given small amount ϵ. The learner is allowed to fail at this task, but only with probability less than δ. The difference between L_h and L_*, $diff_D(L_h, L_*)$, is defined as the probability that L_h and L_* differ in their classification of an instance drawn randomly from X according to D.

The learner must represent its hypothesis concept, L_h, in some representation language. Typically, the representation language restricts the set of possible hypotheses to some limited *concept class C*.

Definition 1 *An algorithm A* pac-learns *a concept class C iff* $\forall L_* \in C$ *and all distributions D, A outputs* L_h *such that*

1. $Pr[diff_D(L_h, L_*) \leq \epsilon] \geq 1 - \delta$
2. *A runs in time polynomial in* $1/\epsilon$, $1/\delta$, *n, and the length of the representation of the target concept* L_*.

Definition 2 *A concept class C is* pac-learnable *if there exists a pac-learning algorithm for C.*

[2]The original Valiant model used two oracles, $EX^+()$ and $EX^-()$ using distributions D^+ and D^- over the positive and negative instances respectively. A number of people have adopted the variant described here (eg. [Amsterdam 88b], [Angluin 88], [Haussler 87]). The relationship between the two models appears in [Haussler *et.al* 88].

Many positive and negative results have emerged from this framework. For example, k-DNF and k-CNF are *pac*-learnable [Valiant 84] while k-term-DNF and k-clause-CNF are not (directly) *pac*-learnable unless $RP = NP$ [Pitt & Valiant 88]. Many other results exist, both [Pitt & Valiant 88] and [Haussler 87] provide excellent summaries of these and related results.

3 Bias Evaluating Algorithms

The Valiant model provides some very powerful guarantees on efficiency, accuracy, and reliability, but only when the initial assumption holds that the target concept L_* is a member of the class C. When the assumption is false, the framework itself places no limitations on the behavior of a *pac*-learner. While it may be possible for some specific existing *pac*-learning algorithms to obtain performance guarantees when the bias is incorrect, in general *pacness* alone says nothing about performance in this case. Typically the reliability and accuracy of the final concept will drop below $1 - \delta$ and $1 - \epsilon$ respectively.

An algorithm's bias will be termed *correct* when the assumptions made by the learner about the target concept, from which the algorithm derives its performance guarantees, are true. For a strict *pac*-learning algorithm as defined in the previous section, this simply reduces to the assumption that the target concept belongs to the concept class C as captured by the following definition. In Section 3.3 the definition will be generalized.

Definition 3 *A bias C is* correct *with respect to an algorithm A for a target concept L_* if A* pac*-learns C and $L_* \in C$. Otherwise it is* incorrect.

An extension to the basic framework introduced below makes the Valiant model robust with respect to the correctness of bias. The extension is motivated by a few simple considerations. First, such an extension should not sacrifice the polynomial-time complexity *or* the reliability of a *pac*-learning algorithm since the user will still require in a reasonable amount of time a response that can be believed. However, an incorrect bias could make it computationally difficult, or even impossible, to achieve the desired concept accuracy. Rather than misrepresenting its abilities when the bias is incorrect, the algorithm should admit defeat by reporting a bad bias. Even when the bias is not totally correct, there is always a chance that the algorithm might get lucky and stumble upon a sufficient approximation. In such a case it would be preferable to output the approximation rather than reporting the incorrect bias. The following variant of *pac*-learning incorporates these considerations.

Definition 4 *An algorithm A_B is a* bias-evaluating pac-learning algorithm *for a concept class C and a target concept L_* if for all distributions D, A_B either outputs L_h or reports a bad bias such that*

1. *If $L_* \in C$ (ie. the bias is correct) then with probability at least $1 - \delta$, A_B outputs L_h and $diff_D(L_h, L_*) \leq \epsilon$.*
2. *If $L_* \notin C$ (ie. the bias is incorrect) then with probability at least $1 - \delta$, A_B either outputs L_h and $diff_D(L_h, L_*) \leq \epsilon$* or *$A_B$ reports a bad bias.*

3. In all cases, A_B runs in polynomial-time.

The output of a *bias-evaluating pac*-learning algorithm can be regarded as "$1-\delta$ reliable regardless of whether or not the bias is correct. Thus, the bias-evaluation extension solves the deficiency of the Valiant model discussed earlier. Of course, for this extension to be of any use, it is still necessary to show that *bias-evaluating pac*-learning algorithms exist for interesting concept classes.

3.1 Achieving Reliability

It is shown in this section that not only do bias-evaluating *pac*-learning algorithms exist for interesting concept classes, they exist for all *pac*-learnable concept classes[3]. It is possible using a technique described below to convert any existing *pac*-learning algorithm A into one which reliably reports an incorrect bias. The technique consists of appending to the end of A a simple sampling test which is independent of both A and the concept class C. The resulting algorithm will be referred to as A_B.

As with the learning algorithm, the possibility exists that the test might reach a bad conclusion in the unlikely event that it is dealt a bad sample. To achieve the final desired level of accuracy and reliability while allowing for this possibility, A is run at a slightly more demanding level of performance than before. Increasing the accuracy and reliability of A decreases demands for accuracy and reliability on the sampling test. The constant ψ is introduced to account for this tradeoff. The results below hold for any constant ψ strictly between 0 and 1. When constructing a bias-evaluating *pac*-learner from a given *pac*-algorithm A (where the complexity of A is known), ψ can be chosen so as to minimize the total sampling or computational complexities. A complete description of the algorithm A_B is shown in Figure 1.

The resulting algorithm A_B is a reliable bias-evaluating *pac*-learning algorithm. Before proceeding with the proof of this fact (Theorem 1), it will be necessary to introduce a few lemmas. For clarity, a summary of the notation used throughout the remainder of this section is given in Figure 2. The shorthand $\epsilon' = diff_D(L_h, L_*)$ is used for convenience.

Lemma 1 $Pr\left[|\epsilon'_{obs} - \epsilon'| \leq \frac{\epsilon - \epsilon_{new}}{2}\right] \geq 1 - \delta_{new}$

Proof: This follows from Hoeffding's inequality [Hoeffding 63]: $Pr\left[|\overline{X} - \mu| \leq \lambda\right] \geq 1 - 2e^{-2m_T\lambda^2}$

$$Pr\left[|\epsilon'_{obs} - \epsilon'| \leq \frac{\epsilon - \epsilon_{new}}{2}\right] \geq 1 - 2e^{-2\left(\frac{2}{(\epsilon-\epsilon_{new})^2}\ln\frac{2}{\delta_{new}}\right)\left(\frac{\epsilon-\epsilon_{new}}{2}\right)^2} = 1 - 2e^{-\ln\frac{2}{\delta_{new}}} = 1 - \delta_{new} \quad (1)$$

□

Lemma 2 *If* $\epsilon' > \epsilon$ *then* $Pr\left[\epsilon'_{obs} > \frac{\epsilon_{new}+\epsilon}{2}\right] \geq 1 - \delta_{new}$

[3]Provided that concepts in that class can be evaluated in polynomial time. This is a very reasonable assumption as will be discussed.

Given a *pac*-learning algorithm A and a constant $0 < \psi < 1$, the algorithm A_B that accepts as inputs ϵ and δ can be constructed to be as follows.

1. Compute $\epsilon_{new} = \psi\epsilon$, $\delta_{new} = 1 - \sqrt{1-\delta}$, and $m_T \geq \frac{2}{(\epsilon-\epsilon_{new})^2} \ln \frac{2}{\delta_{new}}$.
2. Run $A(\epsilon_{new}, \delta_{new})$ to obtain L_h.
3. Compute ϵ'_{obs} by testing L_h on m_T randomly selected classified instances and finding the fraction of those instances for which L_* and L_h disagree. Note that ϵ'_{obs} is an estimate for $\epsilon' = diff_D(L_h, L_*)$.
4. If $\epsilon'_{obs} > \frac{\epsilon_{new}+\epsilon}{2}$ then report a bad bias.
5. If $\epsilon'_{obs} \leq \frac{\epsilon_{new}+\epsilon}{2}$ then output L_h.

Figure 1: Construction of Algorithm A_B

A	: The given *pac*-learning algorithm.
L_*	: The target concept.
L_h	: The concept returned by A.
$1-\epsilon$	: Desired accuracy for final concept.
$1-\delta$	: Desired reliability of final result.
$1-\epsilon_{new}$	: The accuracy requested of A by A_B.
$1-\delta_{new}$	: The reliability requested of A by A_B.
ϵ'	: The actual difference between L_h and L_* ($\epsilon' = diff_D(L_h, L_*)$).
ϵ'_{obs}	: The sampled value for ϵ' during A_B's sampling test of L_h.

Figure 2: Notation

Proof:

$$Pr\left[\epsilon'_{obs} > \frac{\epsilon_{new}+\epsilon}{2}\right] \geq Pr\left[\epsilon'_{obs} - \epsilon' > \frac{\epsilon_{new}+\epsilon}{2} - \epsilon\right] = Pr\left[\epsilon'_{obs} - \epsilon' \geq \frac{\epsilon_{new}-\epsilon}{2}\right] \quad (2)$$

$$\geq Pr\left[|\epsilon'_{obs} - \epsilon'| \leq \frac{\epsilon - \epsilon_{new}}{2}\right]$$

$$\geq 1 - \delta_{new} \quad (3)$$

Line 2 is a result of $\epsilon' > \epsilon$, and line 3 follows from Lemma 1. □

Lemma 3 *If* $\epsilon' \leq \epsilon_{new}$ *then* $Pr\left[\epsilon'_{obs} \leq \frac{\epsilon_{new}+\epsilon}{2}\right] \geq 1 - \delta_{new}$

Proof:

$$Pr\left[\epsilon'_{obs} \leq \frac{\epsilon_{new}+\epsilon}{2}\right] \geq Pr\left[\epsilon'_{obs} - \epsilon' \leq \frac{\epsilon_{new}+\epsilon}{2} - \epsilon_{new}\right] = Pr\left[\epsilon'_{obs} - \epsilon' \leq \frac{\epsilon - \epsilon_{new}}{2}\right] \quad (4)$$

$$\geq Pr\left[|\epsilon'_{obs} - \epsilon'| \leq \frac{\epsilon - \epsilon_{new}}{2}\right]$$

$$\geq 1 - \delta_{new} \quad (5)$$

Line 4 follows from $\epsilon' \leq \epsilon_{new}$ and line 5 follows from Lemma 1. □

Theorem 1 *The algorithm* A_B *constructed from* <u>any</u>[4] *pac-learning algorithm A as in Figure 1 is a bias-evaluating pac-learning algorithm. In other words,*

1. *When its bias is correct, with probability at least* $1-\delta$, A_B *outputs* L_h *and* $diff_D(L_h, L_*) \leq \epsilon$.
2. *When its bias is incorrect, with probability at least* $1-\delta$, A_B *either outputs* L_h *and* $diff_D(L_h, L_*) \leq \epsilon$ *or* L_h *reports an incorrect bias.*
3. A_B *runs in polynomial time*[4].

Proof: The first subpart follows from Lemma 3 as shown here. Since A is a *pac*-learning algorithm with a correct bias, $Pr[\epsilon' \leq \epsilon_{new}] \geq 1 - \delta_{new}$. A_B will only output L_h when $\epsilon'_{obs} \leq \frac{\epsilon_{new}+\epsilon}{2}$, and L_h is only correct when $\epsilon' \leq \epsilon$. For A_B to output a correct pac_ϵ-concept, two conditions must be met:

1. A_B outputs L_h — ie. $\epsilon'_{obs} \leq \frac{\epsilon_{new}+\epsilon}{2}$.
2. L_h is pac_ϵ-correct — ie. $\epsilon' \leq \epsilon$.

Therefore, the probability that A_B outputs a pac_ϵ-correct concept is

$$Pr\left[\epsilon' \leq \epsilon \wedge \epsilon'_{obs} \leq \frac{\epsilon_{new}+\epsilon}{2}\right] \geq Pr\left[\epsilon' \leq \epsilon_{new} \wedge \epsilon'_{obs} \leq \frac{\epsilon_{new}+\epsilon}{2}\right] \quad (6)$$

$$\geq Pr[\epsilon' \leq \epsilon_{new}] \cdot Pr\left[\epsilon'_{obs} \leq \frac{\epsilon_{new}+\epsilon}{2} \,\middle|\, \epsilon \leq \epsilon_{new}\right]$$

$$\geq (1 - \delta_{new})(1 - \delta_{new}) = 1 - \delta \quad (7)$$

Line 6 follows from $\epsilon_{new} < \epsilon$ an line 7 follows from A's *pacness* and Lemma 3.

The second subpart of the theorem is proven by considering the following three mutually disjoint and exhaustive cases:

[4]As long as A always runs in polynomial-time and the concepts output by A can be evaluated in polynomially-bounded time.

1. $\epsilon'_{obs} > \frac{\epsilon_{new}+\epsilon}{2}$ — Bias incorrect.
2. $\epsilon'_{obs} \leq \frac{\epsilon_{new}+\epsilon}{2} \wedge \epsilon' \leq \epsilon$ — Correct *pac*-concept.
3. $\epsilon'_{obs} \leq \frac{\epsilon_{new}+\epsilon}{2} \wedge \epsilon' > \epsilon$ — Incorrect *pac*-concept.

Consider case 3 first. From Lemma 2:

$$Pr\left[\epsilon'_{obs} \leq \frac{\epsilon_{new}+\epsilon}{2} \wedge \epsilon' > \epsilon\right] \leq Pr\left[\epsilon'_{obs} \leq \frac{\epsilon_{new}+\epsilon}{2} \middle| \epsilon' > \epsilon\right] \leq \delta_{new} \tag{8}$$

We are interested in the probability of being in one of the first two cases, and since $\delta_{new} < \delta$, it follows that

$$Pr[\text{Outputs "Bias Incorrect" } \underline{\text{or}} \text{ Outputs } pac_\epsilon\text{-correct concept}] \geq 1 - \delta_{new} \geq 1 - \delta \tag{9}$$

Finally, since A runs in time polynomial in $1/\epsilon_{new}$ and $1/\delta_{new}$, and since $1/\epsilon_{new}$ and $1/\delta_{new}$ are themselves polynomial in $1/\epsilon$ and $1/\delta$, the run time of A is polynomial in $1/\epsilon$ and $1/\delta$. The remainder of A_B, comparing L_h to L_* on the m_T instances to obtain ϵ'_{obs}, requires $O(m_T \cdot evaltime)$ where *evaltime* is the polynomially-bounded amount of time required to evaluate L_h. □

For the polynomial time of Theorem 1 to hold, it must be the case that the concept description for L_h produced by A can be evaluated (ie. used to classify a given instance) in polynomial time. This assumption is an extremely reasonable one since if this was not the case, A itself would not be very useful even when the bias is correct. Also, it is almost universally the case that learning algorithms evaluate concept descriptions at some point during learning. If A does so and is still polynomial, then the concept descriptions examined must also be polynomial.

Since C being a *pac*-learnable concept class implies the existence of a *pac*-learning algorithm for C (Definition 2), and since any *pac*-learning algorithm can be extended to evaluate its bias, the following corollary follows directly from Theorem 1.

Corollary 1 *For any pac-learnable concept class C such that all concepts in C can be evaluated in polynomially-bounded time, there exists a bias-evaluating pac-learning algorithm for C.*

3.2 Another Look at Bias-Evaluation

With the introduction of Bias-Evaluation now complete, it is worthwhile to once again examine Bias-Evaluation at a more intuitive level. In Figure 3a the target concept L_* is in the concept class C; therefore, algorithm A's bias is correct. As a result, with high probability A (and A_B) will return a concept within ϵ of L_*. One can think of a ball of radius ϵ centered at L_* which contains the possible ϵ-approximations of L_*.

A second case where A's bias is blatantly insufficient for approximating L_* is shown in Figure 3b. Not only does L_* not belong to C, *no* concept within ϵ of L_* belongs to C. Clearly A's bias is incorrect and with probability $1 - \delta$ or better A_B will report a bad bias.

The interesting *fuzzy* case appears in Figure 3c. By definition A's bias is incorrect since $L_* \notin C$; however, concepts do exist in C which are sufficient ϵ-approximations to L_*. When the learner does find one of these, it is desirable to return it. This raises the question as to why

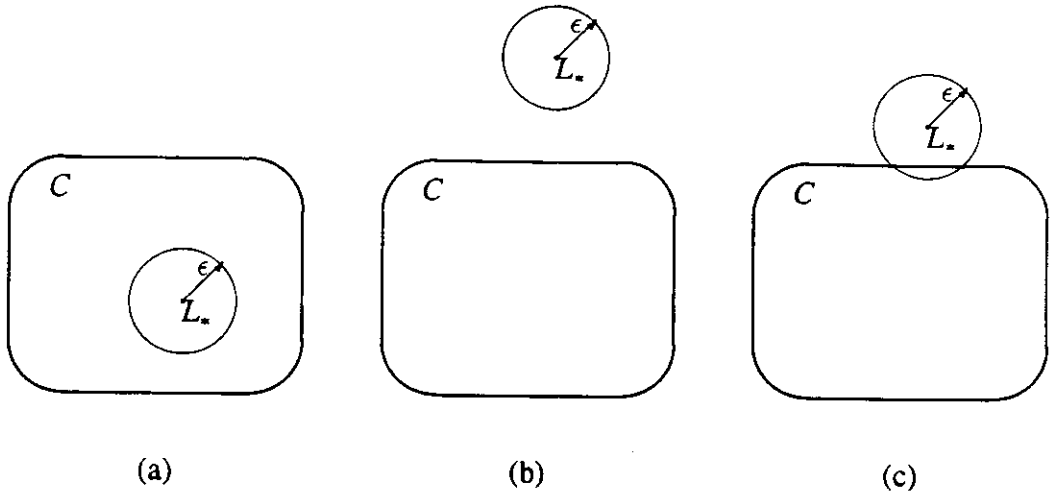

Figure 3:

not require a *pac*-learner to always find an ϵ-approximate concept whenever one exists, even if $L_* \notin C$. There are two reasons. First, since the overlap between C and the ϵ-ball around L_* could be as small as a single concept, the difficulty of the learning task can increase greatly. In many cases this can make an otherwise tractable concept class intractable. Second, without some constraint on L_* such as the knowledge of which concept class it belongs to, it can be very difficult to derive performance guarantees like those afforded by the Valiant framework. Bounds on the computation time and the number of training examples necessary for convergence for existing *pac*-algorithms are possible precisely because it is possible to take advantage of such constraints. From these considerations, it follows that it would be unreasonable to *require* an algorithm to return an ϵ-approximate concept in this case — instead the algorithm should be considered lucky if it can. Otherwise, reporting a bad bias *is* a correct answer. For the total case of Figure 3c, with probability $1 - \delta$ or better, the Bias-Evaluating algorithm A_B will either report a bad bias or return an ϵ-approximation to L_*.

Note that when the bias is correct, A_B can err either by returning a poor concept or by reporting a bad bias. A_B can err when the bias is incorrect only by returning a bad concept. However, in either case, the probability of making an error is less than δ.

3.3 Extending Bias-Evaluation

In this section it is shown how Bias-Evaluation can be applied to an extension of the basic Valiant framework, namely learning in h-dense concept classes [Amsterdam 88a,b]. It is hoped that this example will help to demonstrate how the general idea behind Bias-Evaluation might be applied to other extensions or possibly other frameworks for learning. The example also serves as a comparison of Bias-Evaluation to a closely related work.

In Section 3.2 it was suggested that even when L_* is outside of C, it would be desirable for

a learning algorithm to return an ϵ-approximation to L_* if possible. Both [Kearns *et.al.* 87] with h-heuristic-learnability and [Amsterdam 88a,b] with h-density have addressed this possibility. The h-density extension will be considered in detail here (most of the high level discussion applies to h-heuristic-learning as well).

The results on h-density provide performance guarantees when an algorithm learns within a concept class C_1 but the target concept belongs to a more general concept class C_2. Because L_* is assumed to be in C_2, it is possible to derive guarantees like those from *pac*-learning, even for the case where $L_* \notin C_1$ but $L_* \in C_2$. Therefore, h-density provides one important alternative strategy for coping with the possibility of an incorrect bias. Even so, an h-dense learning algorithm is still biased by the assumption that $L_* \in C_2$ and therefore, Bias-Evaluation can be useful in conjunction with an h-dense learner.

[Amsterdam 88a,b] quantifies the notion of the distance between two concept classes with the parameter h:

Definition 5 (Amsterdam 88a,b) *A concept class C_1 is* h-dense *in a concept class C_2 iff, for any concept $c_2 \in C_2$ and any probability distribution D over the example space, there is a concept $c_1 \in C_1$ such that $diff_D(c_1, c_2) \leq h$.*

Unlike the case of a strict *pac*-learning algorithm in which a user may request an arbitrary close approximation, the user of an h-dense learning algorithm may not request anything as close as or better than an h-approximation to the target concept. For some target concept, this may be the best achievable.

Let A^h be an h-dense *pac*-learning algorithm. Given $0 < \epsilon \leq 1$ and $0 < \delta \leq 1$, A^h outputs a concept L_h from C_1 such that if C_1 is h-dense in C_2

1. A^h runs in polynomial-time.

2. When $L_* \in C_2$, $Pr[diff_D(L_h, L_*) \leq h + \epsilon] \geq 1 - \delta$.

To apply Bias-Evaluation to A_h, it is necessary to modify the definition of *correct* bias for this case. Specifically, the bias is correct when $L_* \in C_2$; however, rather than require a change in the definition for every new extension, the following is a more general definition which subsumes both Definition 3 and the h-density extension. The intuition behind this definition is that the bias is correct when all the assumptions which influence an algorithm's performance guarantees are correct.

Definition 6 *A bias C is correct with respect to an algorithm A for target concept L_* if for any ϵ and δ, A outputs L_h and $Pr[diff_D(L_h, L_*)] \leq \epsilon] \geq 1 - \delta$.*

A quick examination shows that for strict *pac*-learning this definition reduces to Definition 3, and it also reduces to $L_* \in C_2$ above as desired. As before, it is possible to append a Bias-Evaluator to A^h, the resulting algorithm A^h_B is shown in Figure 4. A trivial adjustment to the proof of Theorem 1 yields:

Given an h-dense *pac*-learning algorithm A^h and a constant $0 < \psi < 1$, the algorithm A_B^h that accepts as inputs ϵ and δ can be constructed to be as follows.

1. Compute $\epsilon_{new} = \psi\epsilon$, $\delta_{new} = 1 - \sqrt{1-\delta}$, and $m_T \geq \frac{2}{(\epsilon-\epsilon_{new})^2} \ln \frac{2}{\delta_{new}}$.
2. Run $A^h(\epsilon_{new}, \delta_{new})$ to obtain L_h.
3. Compute ϵ'_{obs} by testing L_h on m_T randomly selected classified instances and finding the fraction of those instances for which L_* and L_h disagree. Note that ϵ'_{obs} is an estimate for $\epsilon' = diff_D(L_h, L_*)$.
4. If $\epsilon'_{obs} > h + \frac{\epsilon_{new}+\epsilon}{2}$ then report a bad bias.
5. If $\epsilon'_{obs} \leq h + \frac{\epsilon_{new}+\epsilon}{2}$ then output L_h.

Figure 4: Construction of Algorithm A_B^h

Theorem 2 *Let C_1 be h-dense in C_2 and let A^h be an h-dense learning algorithm that outputs $L_h \in C_1$. The algorithm A_B^h constructed from A_h as in Figure 4 is an h-Bias-Evaluating algorithm. In other words,*

1. *When $L_* \in C_2$, with probability at least $1-\delta$, A_B^h outputs L_h and $diff_D(L_h, L_*) \leq h + \epsilon$.*
2. *When $L_* \notin C_2$, with probability at least $1-\delta$, A_B^h either outputs L_h and $diff_D(L_h, L_*) \leq h+\epsilon$ or A_B^h reports a bad bias.*
3. *A_B^h runs in polynomial-time[5].*

It is worth examining the relationship between A_B and A_B^h. Given that a program outputs L_h from C_1, and that C_1 is h-dense in C_2, we should compare the behavior of the Bias-Evaluator A_B which utilizes a strict *pac*-learning algorithm with the Bias-Evaluator A_B^h which utilizes an h-dense learning algorithm. The difference is that there is a tradeoff between the level of accuracy that may be requested and the likelihood that the algorithm returns a concept (as opposed to reporting a bad bias) in the event that $L_* \notin C_1$.

One of the motivations behind the original introduction of h-density was its application for dealing with the possibility of a bad bias. When a class C is known to be h-dense in $\mathcal{U}$ (the universal concept class), then h-density by itself provides an alternative to Bias-Evaluation. This can be seen as an extreme of the above tradeoff — the algorithm always returns an $(h+\epsilon)$-approximate concept, but this comes with the price of h being very large. In particular, [Amsterdam 88a] shows that if the *Vapnik-Chervonenkis dimension*[6] of C is n, then C cannot be h-dense in $\mathcal{U}$ for $h < \frac{1}{n+1}$.

[5]Provided A^h always runs in polynomial-time and outputs concepts that can be evaluated in polynomial-time.

[6]See [Haussler 87].

4 Related Work

While the majority of results within the Valiant framework depend on the correct bias assumption, a few people have addressed the possibility of an incorrect bias. The results on h-heuristic learnability [Kearns *et.al* 87] and h-density [Amsterdam 88a,b] are two very noteworthy instances that have already been discussed in Section 3.3.

The approach used in this paper for performing the bias-evaluation is very closely related to *Hypothesis Filtering* [Etzioni 88]. Hypothesis filtering uses the fact that a given hypothesis can be $(1-\delta)$-reliably verified to be $(1-\epsilon)$-accurate by testing it against a random sample of instances. This technique is also discussed in [Kearns *et.al* 87] and was first introduced and used by [Angluin 86]. While bias evaluation and hypothesis filtering are very similar, the objectives are different. Hypothesis filtering was proposed as a method for achieving guaranteed reliability from any learning algorithm by settling for weaker guarantees than the Valiant framework provides. For purposes of comparison, bias evaluation can be viewed as a way of reliably detecting when a learning algorithm should be changed.

In the Machine Learning community, the most notable instance of detecting and dealing with incorrect inductive bias has been the STABB system [Utgoff 86]. Utgoff divides the problem into the issues of detecting a bad bias and switching to a new bias. This paper has addressed only the first issue. STABB detects an insufficient bias when no concept in its hypothesis space is consistent with all observed training instances. Bias evaluation differs from detecting a collapsed version space in a number of notable ways. In some cases, detecting the collapse of a version space might be expensive [Haussler 88], while the addition of bias evaluation to an algorithm is relatively cheap. Bias evaluation may also be slightly more powerful than the collapsing version space approach since in some cases bias evaluation can detect an incorrect bias before the version space collapses, even when the additional testing instances are used for learning[7]; however, it is unclear empirically how often such cases arise. Nevertheless, the most significant difference is that bias evaluation *guarantees* reliable detection while prior work along these lines has detected incorrect bias but has not provided guarantees on the reliability of its mechanisms for doing so.

5 Conclusion

Generally it is not possible to know for certain the correct inductive bias to use for learning a given unknown target concept. Reliably detecting when a learning algorithm is biased incorrectly is therefore an important issue for an inductive learner. This paper has exploited the idea that one method for detecting an incorrect bias is to take advantage of an algorithm that makes

[7]This is because the bias test is independent of the learning session and concept space complexity. For consistent hypothesis finders, the effect of a training instance is to reduce the size of the version space while the effect of a testing instance is to influence the estimate for the accuracy of a concept L_h. While in the first case an instance might eliminate L_h from the VS, the VS may still contain many other candidate concepts, while in the testing case the instance simply lends support to the possibility that L_h was not as accurate as promised. This effect is most pronounced when the VS is still very large after the learner has (supposedly) obtained ϵ-convergence. The effect appears to be very dependent on the particular learning situation.

strong performance guarantees when the bias *is* correct, then verify that the algorithm performs as promised. The Valiant framework provides a source for such algorithms, and the result from the method are algorithms with strong performance guarantees when the bias is correct as well as when the bias is not correct. The paper has demonstrated that all *pac*-learnable concept classes have reliable bias-evaluating pac-algorithms that run in polynomial time (when the concepts in the class can be evaluated efficiently) by showing how any existing *pac*-learning algorithm can be converted into a bias-evaluating *pac*-learning algorithm.

Acknowledgements

This work has benefited tremendously from many discussions with Haym Hirsh, Oren Etzioni, and my advisor Tom Mitchell. I am grateful to each of them for their valuable input and encouragement.

References

1. Amsterdam, J., "The Valiant Learning Model: Extensions and Assessment." *Masters Thesis*, M.I.T., 1988.

2. Amsterdam, J., "Extending the Valiant Learning Model." *Proc. 5th Int. Conf. on Machine Learning*; 1988.

3. Angluin, D., "Learning Regular Sets from Queries and Counter-Examples." Technical Report TR-464. Dept. of Computer Science, Yale University; 1986.

4. Angluin, D., "Queries and Concept Learning." *Machine Learning* **2** (4); 1988.

5. Etzioni, O., "Hypothesis Filtering: A Practical Approach to Reliable Learning." *Proceedings of the Fifth International Workshop on Machine Learning*; June 1988.

6. Haussler, D., "Applying Valiant's Learning Framework to AI Concept Learning Problems." Technical Report UCSC-CRL-87-11, Dept. of Computer Science, University of California at Santa Cruz; Sept. 1987.

7. Haussler, D., "Quantifying Inductive Bias: AI Learning Algorithms and Valiant's Learning Framework." Artificial Intelligence **36**; 1988.

8. Haussler, D., M. Kearns, N. Littlestone, M. K. Warmuth, "Equivalence of Models for Polynomial Learnability." Tech. Report UCSC-CRL-88-06, U.C. Santa Cruz; 1988.

9. Hoeffding, W., "Probability Inequalities for Sums of Bounded Random Variables." *Journal of the American Statistical Association*, 58(301); 1963.

10. Kearns, M., M. Li, L. Pitt, L. G. Valiant, "Recent Results on Boolean Concept Learning." *Proc. 4th Int. Conf. Machine Learning*; 1987.

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

12. Pitt, L., L.G. Valiant, "Computational Limitations on Learning from Examples." *Journal of the ACM* Vol.35, No.4; Oct. 1988.

13. Russell, R., B. Grosof, "A Declarative Approach to Bias in Concept Learning," in *AAAI-87*, Seattle, WA; 1987.

14. Utgoff, P. E., "Shift of Bias for Inductive Concept Learning." In Michalski, Carbonell, and Mitchell (eds.), *Machine Learning, An Artificial Intelligence Approach Vol. II*, (eds.); Morgan Kaufmann Publishers; 1986.

15. Valiant, L., "A Theory of the Learnable," In *Communications of the ACM*, Vol. 27, No. 11; 1984.