Oblivious PAC Learning of Concept Hierarchies

Michael J. Kearns
AT&T Bell Laboratories
600 Mountain Avenue, Room 2A-423
Murray Hill, New Jersey 07974-0636
mkearns@research.att.com

Abstract

In this paper we introduce an extension of the Probably Approximately Correct (PAC) learning model to study the problem of learning *inclusion hierarchies* of concepts (sometimes called *is-a* hierarchies) from random examples. Using only the hypothesis representations output over many different runs of a learning algorithm, we wish to reconstruct the partial order (with respect to generality) among the different target concepts used to train the algorithm. We give an efficient algorithm for this problem with the property that each run is *oblivious* of all other runs: each run can take place in isolation, without access to any examples except those of the current target concept, and without access to the current pool of hypothesis representations. Thus, additional mechanisms providing shared information between runs are not necessary for the inference of some nontrivial hierarchies.

Introduction and Motivation

In this paper we introduce an extension of the Probably Approximately Correct (PAC) learning model (Valiant 1984) to study the problem of learning *inclusion hierarchies* of concepts (sometimes called *is-a* hierarchies) from random examples. (In this paper, a *concept* is simply a set, or equivalently, its Boolean characteristic function.) Using the hypothesis representations output over many different runs of a learning algorithm, we wish to reconstruct the partial order (with respect to generality) among the different target concepts used to train the algorithm.

Informally, our intention is to model the ability to not only learn (in the sense of recognition) the abstract concepts *chair* and *furniture*, but to also infer that *all chairs are furniture*. The scenario we have in mind is roughly the following: a learning algorithm L is run many times, and each time is trained on a potentially different target concept f. Each run results in the addition of a new hypothesis representation r to an existing pool. In addition to the usual criterion that each r should provide a good predictive approximation to its corresponding target concept f, we would like the pool of hypothesis representations to effectively support *inclusion tests* between any pair of target concepts. More precisely, given any r_1 and r_2 in the pool (where r_1 was the result of training L on a target concept f_1, and r_2 was the result of training L on f_2), by examining r_1 and r_2 we should be able to determine if $f_1 \supseteq f_2$ (that is, $f_2(x) = 1$ implies $f_1(x) = 1$ for all x), $f_2 \supseteq f_1$, or f_1 and f_2 are incomparable. [1]

We will present an algorithm for this problem that meets the following three conditions:

1. Obliviousness: Every run of the algorithm can take place in complete isolation, and is oblivious of all past and future runs. When being trained on a target concept f, the algorithm needs access only to examples of f. It does not consult the current pool of hypothesis representations or receive examples of other concepts. Thus, all runs of the algorithm are pairwise statistically independent.

2. A Closed System: The pool of hypothesis representations output by the algorithm functions as a closed system. By this we mean that the inclusion test on r_1 and r_2 takes place without any additional sampling of values of f_1 or f_2, and the information required to determine the relationship between f_1 and f_2 is entirely contained in r_1 and r_2.

3. Succinctness of Hypotheses: The learning algorithm does not vacuously satisfy the obliviousness condition by, for instance, storing a large random sample of f as part of its hypothesis representation r and thus letting this random sample become part of the "closed system" of representations. Such an approach would effectively allow each inclusion test to be performed with simultaneous access to examples of both target concepts participating in the test, and would render our first two conditions meaningless. The algorithm explicitly synthesizes from the training data some representation that is considerably more succinct than the training data itself, but that facilitates inclu-

[1] The most natural hypothesis representation r would seem to be simply the representation of some concept h that approximates the corresponding target concept f. While such a representation is clearly sufficient for the standard (predictive) learning problem, we will show that in our model it is *not* sufficient for supporting inclusion tests, but that allowing r to represent *two* concepts (that is, $r = (h, h')$, where h and h' are two different concepts approximating f) *is* sufficient. We will eventually clarify this issue rigorously, but for now will continue to refer to "hypothesis representations" in an abstract way.

sion testing.

It is important to emphasize that we in no way consider these three conditions to be definitive for models of learning with inclusion testing. For instance, we do not wish to dismiss models that allow the current run of a learning algorithm access to the current pool of hypothesis representations, or that allow some limited sampling as part of the inclusion testing process. The main claim we wish to support here is that in many cases of interest, such additional mechanisms are *unnecessary*, and the above criteria can be met by provably efficient and correct algorithms.

One of our primary motivations is the problem of performing precise logical inference using imperfect learned predicates. If we interpret the output of a concept learning algorithm as an empirically acquired logical predicate, then the ability to use these predicates to accurately detect inclusions among the corresponding target concepts can be viewed as a simple form of this problem, since inclusion is equivalent to implication or subsumption. We investigate this connection in greater depth in a section near the end of the paper.

A summary of the paper follows. We begin with an informal discussion of the special problems posed for inclusion testing in the PAC model, and follow this with formal definitions for the PAC model. We then sketch our model of PAC learning with inclusion testing, and give a necessary technical aside that limits the pairs of target concepts for which we may reasonably expect to detect inclusion.

Our first result proves that there is no hope that any PAC learning algorithm will already provide inclusion testing via direct comparison (with respect to $\supseteq$) of the hypothesis concepts, because a single hypothesis concept contains insufficient information. This then leads us to consider learning algorithms that compute *multiple* hypothesis concepts on each run. The main technical result of the paper is an algorithm that uses this approach, and provably provides PAC learning with inclusion testing for any concept class that is closed under intersection (or dually, under union). We present this result over two sections, first sketching the intuition informally for a special case, and then stating the general theorem. We then give a modified version with considerably improved sample complexity, and discuss various ways of reconstructing an entire hierarchy or partial order of inclusions. In the final sections, we return to the more general problems of logical inference using learned predicates, and discuss some of our other results.

Why Use the PAC Model?

The choice of the PAC model as the basis for our study of learning concept hierarchies is significant due to the probabilistic nature of the PAC model. In a model where the examples are generated randomly, it is in general impossible to exactly identify the target concept — a small amount of error is unavoidable. From this small error arises the difficulty in detecting inclusions.

As an example of this difficulty, consider the standard algorithm for learning an unknown target rectangle in the real plane whose sides are parallel to the coordinate axes (we will refer to such rectangles as being *axis-parallel*), where the examples are drawn according to an unknown and arbitrary probability distribution over the plane. (We will return to this problem as a running example throughout the paper). The algorithm takes a sufficiently large set of positive examples of the target rectangle f, and outputs as its hypothesis the most specific axis-parallel rectangle that includes all of these positive examples. While it is well-known that this algorithm meets the criteria of the PAC model (Blumer et al. 1989), consider what happens when we run this algorithm twice *independently* (that is, using an independent random sample for each run), using the *same* target rectangle f and the same distribution for each run. Although each of the two hypothesis rectangles h and h' output by the algorithm will be (with high probability) good approximations to f with respect to the distribution, the variations in the random samples for the two runs will almost certainly result in both $(h - h')$ and $(h' - h)$ being nonempty. Thus, it will be impossible to detect that the target was identical for the two runs simply by direct comparison (with respect to the ordering $\supseteq$) of the rectangles h and h'. This example is quite relevant, since detecting when the target was *identical* on two runs is a special (and hardest) case of detecting *inclusions* between targets.

Some additional learning mechanism seems to be required here. In fact, in Theorem 1 we *prove* that some additional mechanism is required in a very general sense. We feel that using the PAC model to study learning hierarchies of concepts captures an interesting question: How is it possible to have imperfect hypotheses for target concepts, yet still be able to use these hypotheses to precisely relate (with high probability) the target concepts to each other with respect to inclusion?

The PAC Learning Model

Let X be any set, and let $\mathcal{F}$ be any class of concepts (Boolean functions) over X. We think of X as the *input space*, and $\mathcal{F}$ as the class of possible *target concepts*. In this paper we will think of concepts both as sets and as functions, and will use the notation $f(x)$ to indicate the $\{0,1\}$ value assigned to x by f, and $f_1 \supseteq f_2$ to indicate that for all $x \in X$, $f_2(x) = 1 \Rightarrow f_1(x) = 1$.

Let $\mathcal{D}$ be any probability distribution over X. On an execution using a particular target concept $f \in \mathcal{F}$ and the distribution $\mathcal{D}$, a learning algorithm in our model will be given access to an oracle $EX(f, \mathcal{D})$ that runs in unit time and returns examples of the form $(x, f(x))$, where x is chosen randomly and independently according to $\mathcal{D}$ (denoted $x \in \mathcal{D}$).

Given any concept h, there is a natural measure of the error of h with respect to f and $\mathcal{D}$, defined by $error(h) = \mathbf{Pr}_{x \in \mathcal{D}}[f(x) \neq h(x)]$. Note that $error(h)$ has an implicit dependence on f and $\mathcal{D}$ that we omit for brevity. If $error(h) \leq \epsilon$, we say that h is *ϵ-good* (with respect to f and $\mathcal{D}$).

We are now ready to give the definition of PAC learning (Valiant 1984) (see also (Haussler et al. 1991)):

Definition 1 *Let $\mathcal{F}$ be any class of concepts over X. Then $\mathcal{F}$ is* efficiently PAC learnable *if there is an algorithm L such that for any target concept $f \in \mathcal{F}$, for any*

distribution $\mathcal{D}$ over X, and for any values $0 < \epsilon < \frac{1}{2}$ and $0 < \delta < \frac{1}{2}$, if L is given inputs ϵ and δ and access to $EX(f, \mathcal{D})$, the following two conditions are met:

- *(Efficiency) L runs in time polynomial in the inputs $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$, and*
- *(Learning) L outputs a concept $h \in \mathcal{F}$ [2] that with probability at least $1 - \delta$ satisfies error$(h) \leq \epsilon$. This probability is taken over the random examples returned by $EX(f, \mathcal{D})$, and any internal randomization of L.*

We refer to h as the hypothesis *of L, and we call L a* PAC learning algorithm for $\mathcal{F}$.

Note that the hypothesis of the learning algorithm is "Probably Approximately Correct", hence the name and acronym of the model.

It will often be the case that the input space X and the class $\mathcal{F}$ have infinite cardinality, but are naturally parameterized by some *complexity measure* n. A typical example is $X_n = \{0,1\}^n$, and $\mathcal{F}_n$ is the class of all conjunctions of literals over the Boolean variables $x_1, \ldots, x_n$. Now let $X = \cup_{n=1}^{\infty} X_n$ and $\mathcal{F} = \cup_{n=1}^{\infty} \mathcal{F}_n$. To study the asymptotic complexity of learning in such cases, we allow the learning algorithm to also have a polynomial dependence on the smallest n such that $f \in \mathcal{F}_n$ in the efficiency condition of Definition 1 above.

PAC Learning with Inclusion Testing

Ideally, we would like to find PAC learning algorithms that meet the following additional criteria: if the learning algorithm is run once using the oracle $EX(f_1, \mathcal{D})$ for $f_1 \in \mathcal{F}$ to obtain the hypothesis $h_1 \in \mathcal{F}$, and then run again *independently* using the oracle $EX(f_2, \mathcal{D})$ for $f_2 \in \mathcal{F}$ to obtain the hypothesis $h_2 \in \mathcal{F}$, then h_1 and h_2 satisfy (with high probability) (1) $h_1 \supseteq h_2$ if and only if $f_1 \supseteq f_2$, and (2) $h_2 \supseteq h_1$ if and only if $f_2 \supseteq f_1$. Thus, we wish to detect any inclusions that hold among pairs of target concepts by direct comparison of the hypotheses output by a PAC learning algorithm.

We will shortly see (Theorem 1) that this demand is actually too ambitious for even the simplest classes $\mathcal{F}$. We will then show that if we allow each run of the learning algorithm to output *two* hypothesis concepts, so that the run on $EX(f_1, \mathcal{D})$ will result in a pair of concepts $(h_1^1, h_1^2) \in \mathcal{F} \times \mathcal{F}$, and the run on $EX(f_2, \mathcal{D})$ will result in a pair of concepts $(h_2^1, h_2^2) \in \mathcal{F} \times \mathcal{F}$, then criteria equally useful to the conditions (1) and (2) above can be achieved. Looking ahead, the intuitive idea is to find two hypothesis concepts that are upper and lower bounds on the target concept with respect to generality (although this too turns out to be difficult, and our algorithm essentially "fakes" this method).

There are two important points to be made regarding the model we have sketched. First, note that although the target concept changes from run to run of the learning algorithm, the underlying distribution $\mathcal{D}$ is the same for all runs. Since we think of $\mathcal{D}$ and representing the learner's (fairly) constant environment, we find this invariance realistic and natural. For instance, it is reasonable to expect that the distribution on chairs used to train the learner is the same as the *restriction* of the distribution on furniture *to* chairs, and it can even be argued that this invariance is what makes detection of inclusions possible at all. It is also crucial to the results we present here.

Second, notice that we have limited our attention to detecting inclusions only between pairs of concepts. This will be justified later in the paper, where we provide several different methods of reconstructing an entire hierarchy (or more generally, a partial order) using only pairwise inclusion tests.

A Degenerate Case of Non-Inclusion

Before presenting our results, we first need to place a technical restriction on the pairs (f_1, f_2) for which we demand correct inclusion detection. Returning to our running example will best serve our purposes here: suppose that f_1 and f_2 are overlapping axis-parallel rectangles in the real plane such that $f_1 \not\supseteq f_2$ (thus $f_2 - f_1 \neq \emptyset$), but $\mathcal{D}[f_2 - f_1] = 0$. Then the distribution of labeled examples generated by the oracle $EX(f_2, \mathcal{D})$ is *identical* to the distribution of labeled examples generated by the oracle $EX(f_2', \mathcal{D})$, where $f_2' = f_1 \cap f_2$ (also an axis-parallel rectangle). Thus, the learning algorithm has no hope of distinguishing whether the target rectangle is f_2 or f_2' (even given unbounded sampling and computation time), and the answer to inclusion tests with f_1 is different in each case. This motivates the following definition.

Definition 2 *Let f_1 and f_2 be concepts from the class $\mathcal{F}$ over X, and let $\mathcal{D}$ be a distribution on X. For $0 < \gamma < \frac{1}{2}$, we say that the pair (f_1, f_2) is* γ-fair *(with respect to $\mathcal{D}$) if the following two conditions hold:*

- *either $f_1 \supseteq f_2$ or $\mathcal{D}[f_2 - f_1] \geq 2\gamma$, and*
- *either $f_2 \supseteq f_1$ or $\mathcal{D}[f_1 - f_2] \geq 2\gamma$.*

For the reasons given above, for the remainder of the paper we will restrict our attention to the problem of detecting inclusions between pairs of target concepts (f_1, f_2) that are ϵ-fair, where ϵ is the error input to the learning algorithm. [3]

Notice that if we only consider ϵ-fair pairs of target concepts, there is a significant statistical separation between inclusion and non-inclusion: for any target concepts $f_1, f_2 \in \mathcal{F}$, either $f_1 \supseteq f_2$ (in which case $\mathcal{D}[f_2 - f_1] = 0$) or $\mathcal{D}[f_2 - f_1] \geq 2\epsilon$. This suggests that if we have perpetual access to the distribution $\mathcal{D}$ (or alternatively, are willing to store large random samples from $\mathcal{D}$), then we can detect inclusions simply on the basis of an appropriate statistical test. We can indeed prove that this is possible in a rather general sense. Such solutions violate the three conditions outlined in the Introduction and Motivation section. We will proceed to describe algorithms avoiding such violations.

[2] A more standard definition allows h to be a member of a possibly more expressive hypothesis class $\mathcal{H} \supseteq \mathcal{F}$. However, the given definition suffices for the purposes of this paper.

[3] We choose to make double use of ϵ in this way only for simplicity. We could instead add a new input γ to the learning algorithm, demanding that all pairs be γ-fair and allowing the running time to depend polynomially on $\frac{1}{\gamma}$.

PAC Algorithms Do Not Suffice

The obvious approach to inclusion testing is the one we have already sketched: suppose L is a PAC learning algorithm. If L outputs a concept h_1 in response to training on f_1, and L outputs a concept h_2 in response to training on f_2, then to determine if $f_1 \supseteq f_2$, just see if $h_1 \supseteq h_2$. Earlier we showed that for a *particular* algorithm for PAC learning rectangles in the plane, this approach encounters difficulty. Our first result shows that this approach in fact fails for essentially *any* concept class and *any* PAC algorithm.

Theorem 1 *Let $\mathcal{F}$ be any class of concepts over X containing at least two concepts that are different but not complementary. Then there is no PAC learning algorithm L for $\mathcal{F}$ meeting the following condition: For any $\mathcal{D}$ and any pair $(f_1, f_2) \in \mathcal{F} \times \mathcal{F}$ that is ϵ-fair with respect to $\mathcal{D}$, if L is run using $EX(f_1, \mathcal{D})$ to obtain a hypothesis concept h_1, and L is run using $EX(f_2, \mathcal{D})$ to obtain a hypothesis concept h_2, then with probability at least $1 - \delta$, $h_1 \supseteq h_2$ if and only if $f_1 \supseteq f_2$, and $h_2 \supseteq h_1$ if and only if $f_2 \supseteq f_1$.*

Proof: Suppose for contradiction that L meets the stated condition. Let $m(\epsilon, \delta)$ be the number of examples drawn by L on inputs ϵ and δ. Without loss of generality (Haussler et al. 1991), we assume that L is deterministic.

Let $\mathcal{D}$ be any fixed distribution on X. Fix the accuracy input ϵ and the confidence input $\delta \leq \frac{1}{4}$, let $m = m(\epsilon, \delta)$, and define the mapping $G_{L,\mathcal{D}} : \mathcal{F} \to 2^X$ as follows: for any $f \in \mathcal{F}$, $G_{L,\mathcal{D}}(f)$ is the unique concept over X such that when L is run with inputs ϵ and δ and access to oracle $EX(f, \mathcal{D})$, the hypothesis concept h output satisfies $h = G_{L,\mathcal{D}}(f)$ with probability at least $1 - \delta$. To see that $G_{L,\mathcal{D}}$ is well-defined, notice that if over two independent runs of L using oracle $EX(f, \mathcal{D})$, the probability exceeds δ that the hypotheses h_1 and h_2 output by L do not satisfy $h_1 = h_2$, then the probability that (h_1, h_2) fails to meet the conditions of the theorem for the pair (f, f) exceeds δ.

We now give a lemma proving that the output of L must remain constant under certain small perturbations of the target distribution.

Lemma 2 *Let $x \in X$, let $\mathcal{D}_0$ be a distribution on X, and let $\mathcal{D}_1$ be the distribution that is generated by drawing from $\mathcal{D}_0$ with probability $1 - \frac{1}{2m}$ and drawing x with probability $\frac{1}{2m}$. Then for all $f \in \mathcal{F}$, $G_{L,\mathcal{D}_1}(f) = G_{L,\mathcal{D}_0}(f)$.*

Proof: The probability that in m draws from $\mathcal{D}_1$ all m points are drawn from $\mathcal{D}_0$ is at least $\frac{1}{2}$. Thus the probability that L outputs a hypothesis concept h such that $h = G_{L,\mathcal{D}_0}(f)$ in m draws from $EX(f, \mathcal{D}_1)$ is at least $\frac{1}{2}(1 - 2\delta) = \frac{1}{2} - \delta > \delta$ for $\delta \leq \frac{1}{4}$. This implies $G_{L,\mathcal{D}_1}(f) = G_{L,\mathcal{D}_0}(f)$. ■(Lemma 2)

Now let $f_1, f_2 \in \mathcal{F}$ and $x \in X$ be such that $x \in f_1 \Delta f_2$ and $X - f_1 \Delta f_2$ is nonempty (these must exist by our assumptions on $\mathcal{F}$). Without loss of generality, we will assume $x \in f_1$ and $x \notin f_2$. Let $\mathcal{D}_0$ be any distribution such that $\mathcal{D}_0[f_1 \Delta f_2] = 0$. Note that with respect to such a distribution, f_1 and f_2 are indistinguishable, so $G_{L,\mathcal{D}_0}(f_1) = G_{L,\mathcal{D}_0}(f_2)$. Assume without loss of generality that $x \in G_{L,\mathcal{D}_0}(f_1)$.

Now construct a sequence of distributions $\mathcal{D}_1, \ldots, \mathcal{D}_t$ such that for each $1 \leq i \leq t$, $\mathcal{D}_i$ is generated by drawing from $\mathcal{D}_{i-1}$ with probability $1 - \frac{1}{2m}$, and drawing x with probability $\frac{1}{2m}$. Then by repeated application of Lemma 2, we obtain $G_{L,\mathcal{D}_0}(f_1) = G_{L,\mathcal{D}_0}(f_2) = G_{L,\mathcal{D}_1}(f_2) = \cdots = G_{L,\mathcal{D}_t}(f_2)$, and thus $x \in G_{L,\mathcal{D}_t}(f_2)$. Thus the error of $G_{L,\mathcal{D}_t}(f_2)$ with respect to f_2 and $\mathcal{D}_t$ (which is just $\mathcal{D}_t[G_{L,\mathcal{D}_t}(f_2) \Delta f_2]$) is at least $\mathcal{D}_t[x] = \frac{1}{2m}\sum_{i=0}^{t}(1 - \frac{1}{2m})^i = 1 - (1 - \frac{1}{2m})^{t+1}$. This can be made as close to 1 as desired for t large enough, contradicting the claim that L is a learning algorithm. ■(Theorem 1)

Note that Theorem 1 holds even if the learning algorithm is given unbounded computation time.

The Multiple Hypothesis Method

Theorem 1 shows that there is no hope of simply running a PAC algorithm and hoping that direct comparison (with respect to $\supseteq$) of the independently computed hypothesis concepts will provide accurate inclusion testing. In this section we begin the description of our approach to providing inclusion testing in a PAC setting by computing *two* hypothesis concepts on each independent run, rather than a single hypothesis.

We begin with a theorem proving that if inclusion *fails* to hold between a pair of ϵ-fair target concepts, inclusion must also fail to hold for any ϵ-good hypotheses.

Theorem 3 *Let f_1 and f_2 be concepts over X satisfying $f_1 \not\supseteq f_2$, and let $\mathcal{D}$ be a distribution over X such that $\mathcal{D}[f_2 - f_1] \geq 2\epsilon$ (thus, (f_1, f_2) is ϵ-fair with respect to $\mathcal{D}$). If h_1 is a concept over X that is ϵ-good with respect to f_1 and $\mathcal{D}$, and h_2 is a concept over X that is ϵ-good with respect to f_2 and $\mathcal{D}$, then $h_1 \not\supseteq h_2$.*

Proof: Suppose for contradiction that $h_1 \supseteq h_2$. Then on each point $x \in (f_2 - f_1)$, either $x \in h_1 \Delta f_1$ or $x \in h_2 \Delta f_2$, so $\mathcal{D}[f_1 \Delta h_1] + \mathcal{D}[f_2 \Delta h_2] \geq \mathcal{D}[f_2 - f_1] \geq 2\epsilon$. Thus we must have either $\mathcal{D}[f_1 \Delta h_1] \geq \epsilon$ or $\mathcal{D}[f_2 \Delta h_2] \geq \epsilon$, a contradiction. ■(Theorem 3)

Stated in the contrapositive, Theorem 3 tells us that if we can find *any* ϵ-good hypothesis h_1 for f_1 and *any* ϵ-good hypothesis h_2 for f_2 such that $h_1 \supseteq h_2$, then we may assert with confidence that $f_1 \supseteq f_2$. This suggests the following approach to designing learning algorithms that support inclusion testing in the PAC model: when training on a target concept f, rather than trying to find a single ϵ-good hypothesis h, the learning algorithm should try to find many "different" hypotheses $(h^1, \ldots, h^k)$, each one ϵ-good for f. Then given the hypothesis list $(h_1^1, \ldots, h_1^k)$ from a run on f_1 and the hypothesis list $(h_2^1, \ldots, h_2^k)$ from a run on f_2, we assert that $f_1 \supseteq f_2$ if and only if for some $1 \leq i, j \leq k$, we have $h_1^i \supseteq h_2^j$. If all hypotheses are ϵ-good with respect to the corresponding target, Theorem 3 guarantees that no false inclusions will be detected; it remains to determine conditions on the hypothesis lists that will guarantee that all true inclusions *will*

be detected. We call this generic scheme the *multiple hypothesis method.*

Theorem 1 proves that the $k = 1$ case of the multiple hypothesis method must always fail. We will shortly prove that for many concept classes of interest, the $k = 2$ case can be made to work.

The intuition is that we would like to use two hypothesis concepts to upper and lower bound the target concept. Thus, when learning f we should find one ϵ-good hypothesis h^s that is more specific than f (that is, $h^s \subseteq f$), and another ϵ-good hypothesis h^g that is more general than f (that is, $h^g \supseteq f$). Our hypothesis list will then be (h^s, h^g). Note that if $f_1 \supseteq f_2$, then we have $h_1^g \supseteq f_1 \supseteq f_2 \supseteq h_2^g$, so the inclusion will be detected correctly by the comparison method for the lists (h_1^s, h_1^g) and (h_2^s, h_2^g) given above. The hypotheses h_1^s and h_2^g are not used to answer the test $f_1 \supseteq f_2$, but are used for the test $f_2 \supseteq f_1$.

Unfortunately, for almost every class of interest (including all those mentioned in this paper) it is difficult to find one of h^s and h^g. As an example of the difficulty, for Boolean conjunctions the set $\mathcal{G}(S)$ of maximally general conjunctions consistent with a set S of positive and negative examples can grow exponentially in the number of variables (Haussler 1988). [4]

However, in the next section we show that for many concept classes, while it may be difficult to find a suitable h^g (or h^s) we can nevertheless successfully *simulate* the above approach.

The Intuition Behind the Algorithm

The idea is to replace the idealized h^s and h^g described above with a "weak" hypothesis h^{wk} and a "strong" hypothesis h^{st}. The names are a reference to the number of examples used to obtain each hypothesis, as will become clear shortly. We think of h^{wk} as playing the role of h^s, and h^{st} as playing the role of h^g. As before, h^{wk} will be more specific than the target concept f, that is $h^{wk} \subseteq f$. In contrast to the idealized h^g, however, h^{st} will also obey $h^{st} \subseteq f$. Thus, our hypothesis list (h^{wk}, h^{st}) for f will consist of two overly specific hypotheses. The key property we will guarantee is that if $f_1 \supseteq f_2$, and we run our learning algorithm once using f_1 to obtain (h_1^{wk}, h_1^{st}) and again independently using f_2 to obtain (h_2^{wk}, h_2^{st}), then with high probability $h_1^{st} \supseteq h_2^{wk}$. We now explain the intuition behind our algorithm's computation of h_2^{wk} and h_1^{st}.

We return to the example of learning axis-parallel rectangles in the plane to illustrate our ideas. Suppose f_1 and f_2 are rectangles such that $f_1 \supseteq f_2$. Consider first the computation of h_2^{wk}, when $EX(f_2, \mathcal{D})$ is the source of examples. Our algorithm will take a certain number m_{wk} of positive examples of f_2 (where m_{wk} is determined by the analysis), and set h_2^{wk} be the most specific rectangle including all these positive examples (see Figure 1, where each $*$ represents a positive example of f_2 received by the algorithm during its computation of h_2^{wk}).

For the analysis, consider the four rectangular subregions L,R,T and B of f_2 that are respectively flush with the left, right, top and bottom of f_2, as shown in Figure 1. Define these regions to each have weight exactly [5] $\frac{1}{(m_{wk})^2}$ under the distribution $\mathcal{D}$ (thus they may have differing areas). We now wish to argue that $L, R, T, B \subseteq (f_2 - h_2^{wk})$, as shown in Figure 1; thus h_2^{wk} has "missed" a "significant" part of f_2 on all four sides, where by "significant" we mean weight $\frac{1}{(m_{wk})^2}$ with respect to $\mathcal{D}$. To see this, note that the probability that we hit L at least once in m_{wk} examples is at most $\frac{m_{wk}}{(m_{wk})^2} = \frac{1}{m_{wk}}$. Thus the probability we fail to hit L is at least $1 - \frac{1}{m_{wk}}$, in which case $L \subseteq (f_2 - h_2^{wk})$. The same analysis obviously holds for the regions R,T and B.

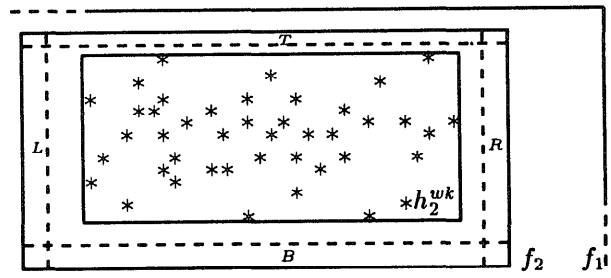

Figure 1.

Now consider our algorithm's independent computation of h_1^{st}, when $EX(f_1, \mathcal{D})$ is the source of examples. With respect to correctly detecting the inclusion $f_1 \supseteq f_2$, the intuitive goal of this computation is to hit all four of the subregions L,R,T and B of f_2 defined above. Note that since each of the subregions has weight exactly $\frac{1}{(m_{wk})^2}$ under $\mathcal{D}$, then if we take $m_{st} \approx (m_{wk})^3$ positive examples of f_1 from $\mathcal{D}$, then with probability considerably greater than $1 - \frac{1}{m_{wk}}$ we will have at least one hit in each of the four subregions. Our hypothesis h_1^{st} is the most specific rectangle including all the positive examples of f_1. If all four subregions were hit on this run, and none of them were hit in the computation of h_2^{wk}, then it is easily verified that $h_1^{st} \supseteq h_2^{wk}$, as desired. These two events occur simultaneously with probability at least $1 - \frac{5}{m_{wk}}$, so for m_{wk} large enough we detect the inclusion with high probability.

The informal analysis given here exploits the geometry of rectangles. In the following section we give our main algorithm, a generalization of the algorithm for rectangles, and state a theorem showing that it works under rather general circumstances, even when this geometric intuition is absent. We emphasize that the sample sizes suggested here were for illustrative purposes only. The proposed algorithm is actually consid-

[4] Note that it is not enough to find any element of both the "lower version space" $\mathcal{S}(S)$ and the "upper version space" $\mathcal{G}(S)$ (Mitchell 1982); here we actually need to know that the chosen elements are *truly* more specific and more general than the target.

[5] Distributions $\mathcal{D}$ that defy such a definition are handled in the full proof; here we simply sketch the intuition.

erably more efficient, and can be further improved as will be shown in a later section.

The Main Algorithm

We first develop the necessary definitions and machinery for our algorithm. Let $\mathcal{F}$ be any class of concepts over X, and let $S \subseteq X$ be any set of positive examples of $f \in \mathcal{F}$. Then we define $min_{\mathcal{F}}(S)$ to be the unique most specific concept in $\mathcal{F}$ that includes the set S — that is, $min_{\mathcal{F}}(S)$ is the concept $f' \in \mathcal{F}$ satisfying $f' \supseteq S$, and for any $f'' \in \mathcal{F}$ such that $f'' \supseteq S$ we have $f'' \supseteq f'$. If no such f' exists, then $min_{\mathcal{F}}(S)$ is undefined.

There is a characterization of the classes for which $min_{\mathcal{F}}(S)$ is always defined due to (Natarajan 1987). We say that $\mathcal{F}$ is *closed under intersection* if for any $f_1, f_2 \in \mathcal{F}$ we have $f_1 \cap f_2 \in \mathcal{F}$.

Theorem 4 *(Natarajan 1987) Let $\mathcal{F}$ be a concept class over X. Then $min_{\mathcal{F}}(S)$ is defined for every set S of positive examples of any $f \in \mathcal{F}$ if and only if $\mathcal{F}$ is closed under intersection, and furthermore we have $min_{\mathcal{F}}(S) = \bigcap_{f' \in \mathcal{F}, f' \supseteq S} f'$.*

We next define the well-known *Vapnik-Chervonenkis dimension* of a class $\mathcal{F}$.

Definition 3 *Let $\mathcal{F}$ be a concept class over X and let S be a finite subset of X. Then S is* shattered *by $\mathcal{F}$ if $|\{f \cap S : f \in \mathcal{F}\}| = 2^{|S|}$.*

Definition 4 *Let $\mathcal{F}$ be a concept class. The* Vapnik-Chervonenkis (VC) dimension *of $\mathcal{F}$ is the largest integer d such that there exists a set S of cardinality d that is shattered by $\mathcal{F}$. If no such d exists, the VC dimension is infinite.*

Our algorithm is defined for any class $\mathcal{F}$ that is closed under intersection and has finite VC dimension, and is given below.

Main Algorithm:

1. Take $m_{wk} = O(\frac{d}{\epsilon} \log \frac{1}{\delta})$ random examples of the unknown target function $f \in \mathcal{F}$ from the oracle $EX(f, \mathcal{D})$, where d is the VC dimension of $\mathcal{F}$. Ignore the negative examples received, and let S^{wk} be the set of positive examples received.
2. Compute $h^{wk} = min_{\mathcal{F}}(S^{wk})$.
3. Take $m_{st} = O(\frac{d^2}{\epsilon\delta} \log^2 \frac{1}{\delta})$ additional random examples from $EX(f, \mathcal{D})$, and let S^{st} be the set of positive examples received.
4. Compute $h^{st} = min_{\mathcal{F}}(S^{wk} \cup S^{st})$.
5. Output (h^{wk}, h^{st}).

Recall our method for answering an inclusion test for targets $f_1, f_2 \in \mathcal{F}$: if the algorithm is run once using target concept f_1 and outputs (h_1^{wk}, h_1^{st}), and run independently using target concept f_2 and outputs (h_2^{wk}, h_2^{st}), then we assert that $f_1 \supseteq f_2$ if and only if $h_1^{st} \supseteq h_2^{wk}$, and assert that $f_2 \supseteq f_1$ if and only if $h_2^{st} \supseteq h_1^{wk}$.

The central theorem we give is the following:

Theorem 5 *Let $\mathcal{F}$ be any concept class over X that is closed under intersection and has VC dimension d. Let $\mathcal{D}$ be any distribution over X, and let $(f_1, f_2) \in \mathcal{F} \times \mathcal{F}$ be ϵ-fair with respect to $\mathcal{D}$. Then if the Main Algorithm is run once using oracle $EX(f_1, \mathcal{D})$ and again independently using oracle $EX(f_2, \mathcal{D})$, then with probability at least $1 - \delta$, we have*

- *(Learning) h_1^{wk} and h_1^{st} will both be ϵ-good with respect to f_1 and $\mathcal{D}$, and h_2^{wk} and h_2^{st} will both be ϵ-good with respect to f_2 and $\mathcal{D}$, and*
- *(Inclusion Testing) $h_1^{st} \supseteq h_2^{wk}$ if and only if $f_1 \supseteq f_2$, and $h_2^{st} \supseteq h_1^{wk}$ if and only if $f_2 \supseteq f_1$.*

Furthermore, if $min_{\mathcal{F}}(S)$ can always be computed in time polynomial in $|S|$, then the Main Algorithm runs in time polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$ and d.

Proof: (Outline) The details of the proof are omitted; here we simply sketch the main ideas. That all four hypotheses are ϵ-good for their corresponding targets follows from a standard PAC analysis; see (Blumer et al. 1989). That no false inclusions will be detected follows immediately from Theorem 3. To show that a true inclusion $f_1 \supseteq f_2$ will be detected, we argue in two steps. First, we must show that the region $f_2 - h_1^{st}$ will have such small weight under $\mathcal{D}$ that m_{wk} examples from $\mathcal{D}$ are unlikely to hit the region. This can be shown with a VC dimension analysis using tools of (Blumer et al. 1989), and a simple probabilistic argument. Second, given that the m_{wk} examples drawn to compute h_2^{wk} miss the region $f_1 - h_1^{st}$, we must show $h_2^{wk} \subseteq h_1^{st}$. This follows from the fact that $\mathcal{F}$ is closed under intersection. ■(Theorem 5)

Note that the Main Algorithm requires only positive examples. There is a straightforward dual to the algorithm that uses only negative examples (in order to find weak and strong maximally *general* hypotheses) that is provably correct for any class closed under union.

Comments and Applications

We begin by noting that our Main Algorithm meets the three conditions given in the Introduction and Motivation section in the strongest possible sense. Obliviousness follows immediately from the fact that each run of the Main Algorithm uses examples of only a single target concept, and is statistically independent of all other runs. The pool of hypothesis pairs (h_i^{wk}, h_i^{st}) clearly form a closed system since we need only consult the two pairs corresponding to target concepts f_i and f_j to determine the relationship between f_i and f_j. The hypothesis pairs are the most succinct representation we could possibly hope for in light of Theorem 1.

It is well-known that $min_{\mathcal{F}}(S)$ can be efficiently computed for the classes of axis-parallel rectangles in n dimensions (Blumer et al. 1989), conjunctions of Boolean literals and k-CNF formulae over the n-dimensional hypercube (Valiant 1984), and subspaces of certain n-dimensional vector spaces (Helmbold, Sloan and Warmuth 1990). To all of these classes we may immediately apply the Main Algorithm and obtain an algorithm running in time polynomial in $\frac{1}{\epsilon}$,

$\frac{1}{\delta}$ and n, thus providing reliable inclusion testing. The same holds for the dual algorithm for the dual classes.

Improving the Sample Complexity

The time and sample complexity of the Main Algorithm are dominated by the strong sample size m_{st}. In this section we describe a method that applies only to certain classes, but greatly reduces m_{st} and hence the complexity of our algorithm. We again illustrate the main idea using the example of axis-parallel rectangles in the plane. Recall that if we have rectangles $f_1 \supseteq f_2$, then the goal in computing h_1^{st} is to hit all four of the regions L,R,T and B (see Figure 1), where each of these four regions had a certain small but significant weight ρ (where $\rho = \frac{1}{(m_{wk})^2}$) under $\mathcal{D}$.

The key observation is if we can somehow make ρ larger while still forcing all four regions to be contained in $f_2 - h_2^{wk}$, then we can make m_{st} smaller, since the regions we are trying to hit now have larger weight under $\mathcal{D}$. To do this, we modify the computation of the weak hypothesis for our Main Algorithm (described here for the target f_2) as follows: starting with $h_2^{wk} = min_{\mathcal{F}}(S^{wk})$ where S^{wk} is the set of positive examples, we "squeeze in" each side of h_2^{wk} until a fraction $\frac{\epsilon}{8}$ of the examples are misclassified on each side (see Figure 2). Note that we are essentially setting $h_2^{wk} = min_{\mathcal{F}}(S')$ for an appropriately chosen subset $S' \subseteq S^{wk}$. Since this results in only $\frac{\epsilon}{2}$ of the examples being misclassified by h_2^{wk}, it is easy to show that h_2^{wk} is still ϵ-good for f_2. Now, however, we may define the regions L,R,T and B so that they have weight $\rho \approx \frac{\epsilon}{16}$. This turns out to be considerably larger than the previous value for ρ, and allows the computation of h_1^{st} in the Main Algorithm (which is left unmodified except for the new sample size) to take only $m_{st} = O(m_{wk})$ examples.

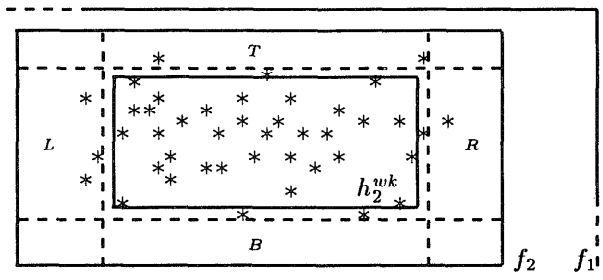

Figure 2.

This modified algorithm is easily generalized to work for the classes of n-dimensional axis-parallel rectangles and conjunctions of Boolean literals over the n-dimensional hypercube. In these cases the required sample sizes now become $m_{wk} = O(\frac{n}{\epsilon}\log\frac{1}{\delta})$ and $m_{st} = O(\frac{n}{\epsilon}\log\frac{1}{\delta})$, so $m_{st} = O(m_{wk})$, and in fact both runs have the optimal number of examples required for standard PAC learning.

Reconstructing a Hierarchy of Concepts

So far we have limited our attention to detecting inclusions between any pair of target concepts. In this section we give methods for reconstructing an entire inclusion hierarchy of target concepts using pairwise inclusion tests among the hypotheses.

Suppose that our Main Algorithm will be called l times on target concepts $f_1, \ldots, f_l$. Then if we set the confidence input on each run to be $\frac{\delta}{l^2}$, where δ is the overall confidence we wish to achieve, it is easy to show that with probability at least $1-\delta$, the inclusion tests between *any* two hypothesis pairs (h_i^{wk}, h_i^{st}) and (h_j^{wk}, h_j^{st}) will correctly determine the inclusion relationship between f_i and f_j.

We have a number of significant improvements to this method that we now describe. First of all, the method as stated has an undesirable dependence on l. In many cases we may not know the value of l in advance. We can in fact give an "on-line" version of this method: if the Main Algorithm is run on a sequence of target concepts $f_1, \ldots, f_i, \ldots$ of *unknown* length, then for any value of i the ith run of the algorithm takes time polynomial in i (and the other usual parameters). The simple trick used is to divide up the confidence parameter δ *gradually* over the runs (using an appropriate decreasing sequence such as $\frac{\delta}{2i\log i}$), rather than allotting an equal portion of δ as is done above. We can also give a reconstruction method in which each run of the Main Algorithm takes time polynomial in the hierarchy depth instead of the size.

Following any of these methods, we can also implicitly propagate positive examples upwards through the hierarchy in a useful way. The basic idea is most easily illustrated for the case of Boolean conjunctions of literals. Suppose that our algorithm has output two hypothesis conjunction pairs (h_1^{wk}, h_1^{st}) and (h_2^{wk}, h_2^{st}) for unknown target conjunctions f_1 and f_2, and we guess that $f_1 \supseteq f_2$ because $h_1^{st} \supseteq h_2^{wk}$. Then assuming this guess is correct, we can do the following: if the variable x_i does not appear in h_2^{st} then we are safe in deleting x_i from h_1^{st} (we will refer to the resulting conjunction as the *modified strong conjunction* for f_1). The reason for this is that if $f_1 \supseteq f_2$, x_i was deleted from h_2^{st} due to a positive example of f_2 in which $x_i = 0$. Since this is also a positive example of f_1, the deletion is justified in h_1^{st} as well.

Thus, starting from the bottom of the hierarchy, we propagate all deletions in the strong conjunction of a node upwards to the strong conjunction of the node's parent, to obtain modified strong conjunctions for all nodes except the leaves. This method has two advantages: first, propagating the deletions upwards results in modified strong conjunctions of greater accuracy; and second, the concept represented by the modified strong conjunction of any node will actually be a subset of the concept represented by the modified strong conjunction of that node's parent. This provides an interesting contrast to Theorem 1, which essentially shows that directly obtaining such conjunctions *as the output of the learning algorithm* is impossible. Here we have shown that such conjunctions can be constructed once all hypothesis pairs are available.

This same method applies to the Main Algorithm: the operation that is analogous to propagating dele-

tions upwards is simply letting the modified strong hypothesis for f_1 be the most specific representation h in $\mathcal{F}$ that satisfies $h \supseteq h_1^{st} \cup h_2^{st}$. Such an h is guaranteed to exist since $\mathcal{F}$ is closed under intersection. [6]

Detecting More Complex Implications

Consider the following straightforward reformulation of the inclusion testing problem as a limited type of logical inference using learned representations: after (independent) training on target concepts f_i and f_j, we would like to use the hypothesis representations r_i and r_j output by a learning algorithm to determine the validity of logical assertions of the form $f_i \Rightarrow f_j$ (shorthand for $(\forall x \in X)[f_i(x) = 1 \Rightarrow f_j(x) = 1]$). Here we are thinking of f_i and f_j as logical predicates whose exact description is inaccessible, and only the *learned* hypothesis representations may be used to determine the validity of the assertion. This reformulation naturally leads us to ask if more complex assertions can also be accurately validated using the methods we have presented. Here we briefly sketch the possibilities and apparent limitations in this direction.

For instance, consider formulae of the form $(\vee_{i=1}^{k} f_i) \Rightarrow f_j$. Such a formula is valid if and only if we have $f_i \subseteq f_j$ for all $1 \leq i \leq k$, so we can use the hypothesis pairs (h_i^{wk}, h_i^{st}) (from independent runs of the Main Algorithm) to guess that $(\vee_{i=1}^{k} f_i) \Rightarrow f_j$ is valid if and only if we have $h_i^{wk} \subseteq h_j^{st}$ for all $1 \leq i \leq k$. It is easy to prove that under our assumption that the target functions are pairwise ϵ-fair with respect to $\mathcal{D}$, this test will with high probability give the correct result for any class $\mathcal{F}$ meeting the conditions of Theorem 5. For similar reasons, we can determine the validity of $f_j \Rightarrow (\wedge_{i=1}^{k} f_i)$ by checking that we have $h_j^{wk} \subseteq h_i^{st}$ for all $1 \leq i \leq k$.

A more subtle example is formulae of the form $f_j \Rightarrow (\vee_{i=1}^{k} f_i)$. When the targets are drawn from a class $\mathcal{F}$ that is closed under intersection, if such a formula is valid we will have (with high probability) $h_j^{wk} \cap f_i \subseteq h_i^{st}$ for each $1 \leq i \leq k$, since $h_j^{wk} \cap f_i \in \mathcal{F}$. From this we obtain $\cup_{i=1}^{k}(h_j^{wk} \cap f_i) \subseteq \cup_{i=1}^{k} h_i^{st}$, or equivalently $h_j^{wk} \cap (\cup_{i=1}^{k} f_i) \subseteq \cup_{i=1}^{k} h_i^{st}$. Since $h_j^{wk} \cap (\cup_{i=1}^{k} f_i)$ is equivalent to h_j^{wk} whenever $f_j \subseteq (\cup_{i=1}^{k} f_i)$ (because $h_j^{wk} \subseteq f_j$ always), this means we can test the validity of the formula by checking that $h_j^{wk} \subseteq (\cup_{i=1}^{k} h_i^{st})$.

The formula type $(\wedge_{i=1}^{k} f_i) \Rightarrow f_j$ is one for which there is no apparent method for using our Main Algorithm's hypothesis pairs to determine validity. Thus it is worth emphasizing that the ability to detect simple implications using learned representations does *not* automatically imply the ability to perform general inference — although our methods handle a number of formula types, they fail on others, and in general we must expect that different hypothesis representations may support or omit various assertion types.

[6] In order to safely apply this method, it is crucial that all pairs of target representations in the hierarchy be ϵ-fair. Otherwise, our algorithm may detect false inclusions, in which case the upward propagation is unjustified and may result in modified strong hypotheses with large error with respect to $\mathcal{D}$.

Other Results and Conclusion

Our Main Algorithm can be modified to apply to some settings in which $min_{\mathcal{F}}(S)$ may not be defined. We can give algorithms that provide accurate inclusion testing for conjunctions of Boolean literals even in the presence of a large rate of classification noise (based on a PAC algorithm due to (Angluin and Laird 1988)), and accurate inclusion testing for monotone DNF formula in the PAC model with membership queries (based on an algorithm due to (Angluin 1988)).

We also have results on applying our methods to detect the validity of formulae in *mixed logic*, where we allow both target function symbols f_i, and *exact* descriptions of *defined* concepts. An example of such a formula is $f_i \Rightarrow (f_j \vee x_1 \overline{x}_3 x_5)$. Here f_i and f_j are target functions whose exact descriptions are Boolean conjunctions (but which we can reason about only via learned hypothesis representations), and $x_1 \overline{x}_3 x_5$ is the exact description of a conjunction over the same variable set.

In conclusion, in this paper we have introduced a new model for studying the detection of inclusions between independently learned target concepts, and have given an algorithm in this model that is efficient, uses minimal shared information between runs, and has optimally succinct hypothesis representations.

Acknowledgements

I would like to give warm thanks to Umesh Vazirani for collaborating in the early stages of this research. I am also grateful to William Cohen, John Denker, Henry Kautz, Rob Schapire and Bart Selman for enjoyable conversations on this material.

References

D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, 2:343–370, 1988.

A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *JACM*, 36(4):929–965, 1989.

D. Haussler, M. Kearns, N. Littlestone, and M. K. Warmuth. Equivalence of models for polynomial learnability. *Information and Computation*, 95(2):129–161, 1991.

D. Helmbold, R. Sloan, and M. Warmuth. Learning integer lattices. In *Proceedings of the 1990 Workshop on Computational Learning Theory*, pages 288–300, San Mateo, CA, 1990. Morgan Kaufmann publisher. Also to appear in *SIAM J. on Computing*.

T. M. Mitchell. Generalization as search. *Art. Intell.*, 18:203–226, 1982.

B. K. Natarajan. On learning boolean functions. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 296–304, New York, New York, May 1987.

L. G. Valiant. A theory of the learnable. *Comm. ACM*, 27(11):1134–42, 1984.