Simulating Access to Hidden Information while Learning

Peter Auer
Institute for Theoretical Computer Science
Technische Universität Graz
Klosterwiesgasse 32/2
A-8010 Graz, Austria

Philip M. Long*
Computer Science Department
Duke University
P.O. Box 90129
Durham, NC 27708 USA

Electronic Colloquium on Computational Complexity, Report No. 67 (2000)

Abstract

We introduce a new technique which enables a learner without access to hidden information to learn nearly as well as a learner with access to hidden information. We apply our technique to solve an open problem of Maass and Turán [18], showing that for any concept class F, the least number of queries sufficient for learning F by an algorithm which has access only to arbitrary equivalence queries is at most a factor of $1/\log_2(4/3)$ more than the least number of queries sufficient for learning F by an algorithm which has access to both arbitrary equivalence queries and membership queries. Previously known results imply that the $1/\log_2(4/3)$ in our bound is best possible. We describe analogous results for two generalizations of this model to function learning, and apply those results to bound the difficulty of learning in the harder of these models in terms of the difficulty of learning in the easier model. We bound the difficulty of learning unions of k concepts from a class F in terms of the difficulty of learning F. We bound the difficulty of learning in a noisy environment for deterministic algorithms in terms of the difficulty of learning in a noise-free environment. We apply a variant of our technique to develop an algorithm transformation that allows probabilistic learning algorithms to nearly optimally cope with noise. A second variant enables us to improve a general lower bound of Turán [19] for the PAC-learning model (with queries). Finally, we show that logarithmically many membership queries never help to obtain computationally efficient learning algorithms.

1 Introduction

We introduce a new and apparently powerful technique which enables a learner to simulate access to hidden information. Such hidden information could be the answer to a query, the knowledge of a value instead of only knowing that a previous prediction was wrong (for learning functions whose ranges have more than two elements), the knowledge whether information is corrupted with noise, et cetera. Clearly all such additional information makes the task of the learner easier, but our results show that learning doesn't become *much* easier.

We introduce our technique by tailoring it to what we think are the most interesting cases. More applications of the technique are given in [3]. Since queries are often substantially more expensive than computation time in practice, in this paper our primary focus is on the number of queries required by learning algorithms.

The paper is organized as follows. In Section 2 we introduce our technique and apply it to compare the complexity of learning with and without arbitrary boolean queries in the on-line learning model. In Section 3 we describe some more applications of the technique. We generalize the main result of Section 2 to two natural models of the learning of functions with arbitrary ranges and compare the difficulty of learning in these models. We compare the difficulty of learning unions of k elements of a concept class with the difficulty of learning the original class. Finally, we give bounds for learning in the presence of noise. In Section 4 we give a nearly optimal randomized algorithm for on-line learning at a high level of noise. In Section 5 we exploit our technique to get lower bounds on the learning complexity in the PAC learning model with additional queries in terms of the Vapnik-Chervonenkis [21] dimension of the class, a well-known measure of the "richness" of the class. Our bound improves on the constant of a recent result of Turán [19], and our proof is somewhat simpler as well. In Section 6, we describe a general bound on the usefulness of boolean queries in designing *computationally* efficient algorithms.

2 On-line learning of concepts with and without boolean queries

In the on-line learning model [1] (also often called the exact learning model) the learner has to learn a function f from some domain X to $\{0,1\}$ (called a *target*

*Supported by Air Force Office of Scientific Research grant F49620–92–J–0515. Most of this work was done while this author was at TU Graz supported by a Lise Meitner Fellowship from the Fonds zur Förderung der wissenschaftlichen Forschung (Austria).

ISSN 1433-8092

concept) from some class $F \subseteq \{0,1\}^X$ (called a *concept class*). The learner's interaction with its environment is modeled with *queries*. To ask an *equivalence query*, the learner proposes a hypothesis $h \in \{0,1\}^X$. If the learner's hypothesis is incorrect, it receives an $x \in X$ which is a counterexample, i.e. which has $h(x) \neq f(x)$. If the hypothesis is correct (i.e., $h = f$), then the learner has learned the target (and this last equivalence query is not counted for measuring the performance of the algorithm). Observe that h need not be from the target class F, in fact *arbitrary* equivalence queries are allowed.[1] A *boolean query* is any subset Q of F, and the learner receives YES if $f \in Q$ and NO otherwise. An example of a boolean query is "Is $f(x) = 1$?" for an x chosen by the learner, often called a *membership query*. A given boolean query $Q \subseteq F$ by an algorithm A *yields information* if there are $f_1 \in Q$ and $f_2 \in F - Q$ such that both f_1 and f_2 are consistent with the answers to the previous queries asked by A. The performance of a deterministic learning algorithm is measured by the number of queries the algorithm needs for learning the most difficult target in F for it. (Randomized learning algorithms will be considered in Section 4.)

In this section we compare the performance of the optimal learning algorithm which asks equivalence queries *and* boolean queries with the performance of the optimal learning algorithm which uses only equivalence queries. (The model in which only equivalence queries are allowed is equivalent to the *mistake-bound* model [14].)

For any concept class F we denote by $\mathrm{opt_{EB}(F)}$ the performance of an optimal learning algorithm for F which uses equivalence and boolean queries and we denote by $\mathrm{opt_E(F)}$ the performance of an optimal learning algorithm for F which uses only equivalence queries. Our first result states that, if the answers to the boolean queries are hidden from the learner, the performance of the learner degrades only by a small constant factor (as $1/\log_2(4/3) \approx 2.41$).

Theorem 2.1 *If $F \subseteq \{0,1\}^X$ then*

$$\mathrm{opt_E(F)} \leq \frac{\mathrm{opt_{EB}(F)}}{\log_2 4/3}.$$

This result solves an open problem posed by Maass and Turán [18]. They proved[2] that $\mathrm{opt_{EB}(F)} \geq \frac{\mathrm{opt_E(F)}}{\log_2(1+\mathrm{opt_E(F)})}$ and asked if there are classes $F = \cup_n F_n$ with $\mathrm{opt_{EB}(F_n)} = \mathrm{o(opt_E(F_n))}$. Our result shows that this is not the case. Angluin and Kharitonov [2] gave several natural examples of specific concept classes for which polynomially many membership queries did not help to obtain *computationally* efficient learning algorithms, modulo cryptographic assumptions.[3]

The VC-dimension [21] of a class F is defined by

$$\mathrm{VCdim(F)} = \max\{\mathrm{d} : \exists \mathrm{x_1}, ..., \mathrm{x_d} \in \mathrm{X}, \{(f(x_1), ..., f(x_d)) : f \in F\} = \{0,1\}^d\}.$$

The fact that $\mathrm{opt_E(F)} \geq \mathrm{VCdim(F)}$ [18] trivially yields the following corollary.

Corollary 2.2 *If $F \subseteq \{0,1\}^X$, then*

$$\mathrm{opt_{EB}(F)} \geq \log_2(4/3)\mathrm{VCdim(F)}.$$

This improves on the $\mathrm{opt_{EB}(F)} \geq \mathrm{VCdim(F)}/7$ bound of Maass and Turán. Furthermore, an example due to Maass and Turán shows that the constant cannot be improved in either Theorem 2.1 or Corollary 2.2. (In their proof, the only boolean queries used are membership queries. Thus, the obvious corollary of Theorem 2.1 bounding the usefulness of membership queries is also optimal up to lower order terms.)

Theorem 2.3 ([18]) *There is a family $\langle X_n \rangle_n$ of sets and a family $\langle F_n \rangle_n$ such that for each n, $F_n \subseteq \{0,1\}^{X_n}$, and*

$$\begin{aligned} \mathrm{opt_{EB}(F_n)} &\leq (\log_2(4/3) + o(1))\mathrm{VCdim(F_n)} \\ &\leq (\log_2(4/3) + o(1))\mathrm{opt_E(F_n)}. \end{aligned}$$

The proof of Theorem 2.1 is based on a technique to simulate access to the answers of the boolean queries without actually receiving these answers.

Proof of Theorem 2.1: *Let A^{EB} be an optimal learning algorithm which for all targets $f \in F$ uses at most $\mathrm{opt_{EB}(F)}$ equivalence and boolean queries. Assume without loss of generality that all of A^{EB}'s boolean queries yield information. We construct a learning algorithm A^{E} which uses at most $\mathrm{opt_{EB}(F)}/\log_2(4/3)$ equivalence queries and no boolean queries.*

The algorithm A^{E} runs copies A_i^{EB} of A^{EB} as subalgorithms and keeps a weight w_i for each copy. Initially A^{E} starts with one copy of A^{EB} and its weight is 1. To prove the theorem we (as observers of the algorithm A^{E}) investigate how the total sum of all weights w_i changes, and we keep track of a special copy A_s^{EB} (and its weight) which performs in the same way as A^{EB} would perform if boolean queries were available. Initially the single copy is the special one. The copies and their weights are maintained by A^{E} in the following way.

- *If some copy A_i^{EB} wants to ask a boolean query Q, this copy is split into two copies, one copy receives the answer* YES *and the other copy receives the answer* NO. *The weight $w_i/2$ is assigned to both copies.*

[1] In a popular variant of this model, the learning algorithm is required to output hypotheses from F.

[2] Their result was only for membership queries but easily generalizes to boolean queries.

[3] In fact, they proved the difficulty of the easier problem of PAC learning with membership queries [20].

Clearly the total sum of weights is not changed.

If A_i^{EB} is the special copy then one of the new copies represents the correct answer to the query and this copy becomes the special one. Its weight is half the weight of the original special copy.

- *When no copy wants to ask a boolean query (because all of A^{EB}'s boolean queries yield information, it is easy to see that eventually this happens), then all copies want to ask equivalence queries, and algorithm A^{E} asks an equivalence query where the hypothesis h is determined by majority vote of the hypotheses h_i of the subalgorithms according to their weights,*

$$h(x) = \begin{cases} 1 & \text{if} \quad \sum_{i:h_i(x)=1} w_i \geq \sum_{i:h_i(x)=0} w_i \\ 0 & \text{if} \quad \sum_{i:h_i(x)=1} w_i < \sum_{i:h_i(x)=0} w_i. \end{cases}$$

If the hypothesis is not correct and A^{E} receives a counterexample x^*, then this counterexample is passed to all subalgorithms which also predicted incorrectly for x^*, i.e. x^* is passed to all A_i^{EB} with $h_i(x^*) = h(x^*)$. Furthermore the weights of all these copies are multiplied by 1/2. The copies that predicted correctly for x^* are not modified and they don't receive a counterexample. They don't change their states and propose the same hypotheses as before.

Since $\sum_{i:h_i(x^*)=h(x^*)} w_i \geq \sum_{i:h_i(x^*)\neq h(x^*)} w_i$, arguing as in [16] we have for the modified weights w_i' that

$$\begin{aligned} &\textstyle\sum_i w_i' \\ &= \textstyle\sum_{i:h_i(x^*)=h(x^*)} w_i' + \sum_{i:h_i(x^*)\neq h(x^*)} w_i' \\ &= \tfrac{1}{2}\textstyle\sum_{i:h_i(x^*)=h(x^*)} w_i + \sum_{i:h_i(x^*)\neq h(x^*)} w_i \\ &= \tfrac{3}{4}\textstyle\sum_i w_i - \tfrac{1}{4}\sum_{i:h_i(x^*)=h(x^*)} w_i \\ &\qquad + \tfrac{1}{4}\textstyle\sum_{i:h_i(x^*)\neq h(x^*)} w_i \\ &\leq \tfrac{3}{4}\textstyle\sum_i w_i. \end{aligned}$$

Thus the total sum of weights decreases by at least a factor 3/4.

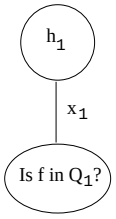
The weight of the special copy is multiplied by 1/2 only if it predicted incorrectly for x^* and receives x^* as counterexample.

To summarize, after M equivalence queries of A^{E} with incorrect hypotheses the total sum of all weights is at most $(3/4)^M$. On the other hand the weight of the special copy is always at least $(1/2)^{\mathrm{opt_{EB}(F)}}$ since the number of equivalence and boolean queries of the special copy is bounded by $\mathrm{opt_{EB}(F)}$. By taking logarithms and solving for M, we get

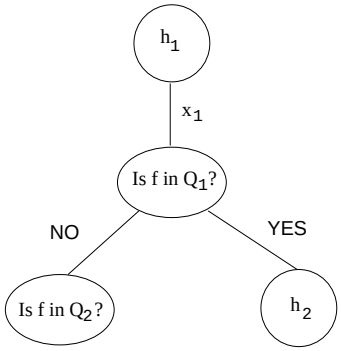
$$M \leq \frac{\mathrm{opt_{EB}(F)}}{\log_2 4/3}$$

which implies the theorem. □

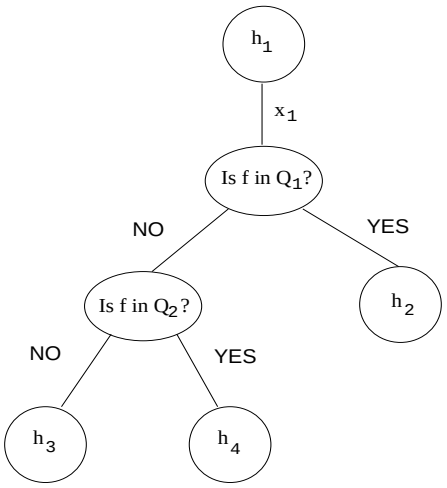
To get a feel for how A^{E} works, it is worthwhile to view its state as a tree, where the various copies of A^{EB} correspond to the leaves. For example, suppose A^{E} is learning f, and at the beginning of A^{E}'s execution, the single copy of A^{EB} would ask an equivalence query h_1. Then the tree at this point would consist of a single node labelled by h_1. Suppose that A^{E} then got a counterexample, call it x_1, which it passed to the single copy of A^{EB}, and that A^{EB} then asked whether f was in some subset Q_1 of F (a boolean query). After this, the tree would look as follows.



Next, A^{E} would create two copies of A^{EB}, one which it would give the response YES, and the other which would get the response NO. If the copy that got the response YES asked another equivalence query, call it h_2, and the copy that got the response NO asked, for some Q_2, whether $f \in Q_2$, then we can visualize the state of A^{E} with the following tree.



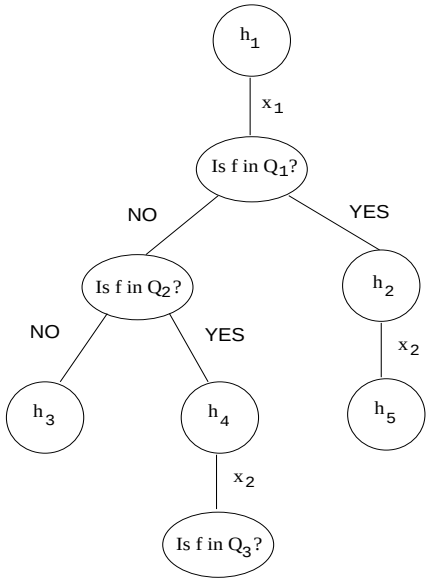
Now, before taking any action on the equivalence query made on the leaf on the right, A^{E} would "expand" the leaf on the left, again creating two copies, which would be given YES and NO respectively as answers to their most recent question. If these copies of A^{EB} each asked equivalence queries, call them h_3 and h_4, then the following tree would encode the state of A^{E}:



Since all the leaves would be asking equivalence queries, A^{E} would be ready to ask an equivalence query, call it g, where, using the weighting scheme in the proof of Theorem 2.1,

$$g(x) = 1 \Leftrightarrow$$
$$h_2(x)/4 + h_3(x)/8 + h_4(x)/8$$
$$\geq (1 - h_2(x))/4 + (1 - h_3(x))/8 + (1 - h_4(x))/8.$$

Suppose the counterexample to g obtained by A^{E}, call it x_2, had $h_2(x_2) = h_4(x_2) = g(x_2)$, and $h_3(x_2) \neq g(x_2)$. Then x_2 would be passed to those copies of A^{EB} whose hypothesis were wrong about it, in this case, to the copies hypothesizing h_2 and h_4. If these copies asked an equivalence query and another membership query respectively, then the new tree would look like this:



The process would continue in this manner, with A^{E} "expanding" all leaves whose copies of A^{EB} ask boolean queries until there are no more such leaves, and then asking an equivalence query constructed from the hypotheses on the leaves as described above.

The above is an example of a more general technique. One may simulate knowledge of hidden information using a tree like the above, "expanding" nodes when the subalgorithm needs information hidden from the master algorithm. The combination of the hypotheses of the leaves is done in the style of Weighted Majority [16], but the main difference with the Weighted Majority algorithm is that the Weighted Majority algorithm uses a fixed set of subalgorithms, while our algorithm dynamically creates subalgorithms depending on previous outcomes to queries.

3 More applications of the basic technique

When generalizing equivalence queries to functions taking on potentially more than two values, there are two very natural alternatives.

In one alternative (called *weak reinforcement* in [4]), the algorithm only discovers some x for which $h(x) \neq f(x)$, but does not find out the value of $f(x)$. This type of reinforcement is sometimes all that is available, for example in some control problems. Let us denote the optimal number of equivalence queries for learning a class $F \subseteq Y^X$ in this model by $\mathrm{opt}_{\mathrm{E,weak}}(\mathrm{F})$. If the algorithm is allowed boolean queries on top of weak equivalence queries, denote the optimal number of queries required for learning F by $\mathrm{opt}_{\mathrm{EB,weak}}(\mathrm{F})$.

In a second alternative (called *strong reinforcement* in [4]), when the algorithm makes an equivalence query h for which $h \neq f$, the algorithm receives an x for which $h(x) \neq f(x)$, and *also receives* $f(x)$. This type of reinforcement also occurs naturally, for example in weather prediction. Let us denote the optimal number of equivalence queries for learning a class $F \subseteq Y^X$ in this model by $\mathrm{opt}_{\mathrm{E,strong}}(\mathrm{F})$. If the algorithm is allowed boolean queries on top of strong equivalence queries, denote the optimal number of queries required for learning F by $\mathrm{opt}_{\mathrm{EB,strong}}(\mathrm{F})$.

Notice that in the case $Y = \{0, 1\}$, we have $\mathrm{opt}_{\mathrm{E}}(\mathrm{F}) = \mathrm{opt}_{\mathrm{E,weak}}(\mathrm{F}) = \mathrm{opt}_{\mathrm{E,strong}}(\mathrm{F})$ and $\mathrm{opt}_{\mathrm{EB}}(\mathrm{F}) = \mathrm{opt}_{\mathrm{EB,weak}}(\mathrm{F}) = \mathrm{opt}_{\mathrm{EB,strong}}(\mathrm{F})$.

The proof of Theorem 2.1 can be modified to obtain generalizations of that result for both alternatives.

Theorem 3.1 *If $F \subseteq Y^X$, then*

$$\mathrm{opt}_{\mathrm{E,strong}}(\mathrm{F}) \leq \frac{\mathrm{opt}_{\mathrm{EB,strong}}(\mathrm{F})}{\log_2 4/3}, \quad (1)$$

$$\mathrm{opt}_{\mathrm{E,weak}}(\mathrm{F}) \leq 1.39|Y|\mathrm{opt}_{\mathrm{EB,weak}}(\mathrm{F}), \quad (2)$$

$$\mathrm{opt}_{\mathrm{E,weak}}(\mathrm{F}) \leq 2^{\mathrm{opt}_{\mathrm{EB,weak}}(\mathrm{F})} - 1. \quad (3)$$

A result similar to (1), but weaker, was proved in [4].

Of course, Theorem 2.3 implies that the constant $1/\log_2 \frac{4}{3}$ of (1) is optimal. The bounds for $\mathrm{opt}_{\mathrm{E,weak}}(\mathrm{F})$

can be seen to be best possible up to a constant factor for certain values of $|Y|$ and $\mathrm{opt}_{\mathrm{EB,weak}}(\mathrm{F})$, as recorded in the following theorem, whose proof is omitted due to space constraints.

Theorem 3.2 *Choose positive integers a and b such that $b \geq 2$. Then there are sets X, Y such that $|Y| = b$, and there is a set F of functions from X to Y such that $\mathrm{opt}_{\mathrm{EB,weak}}(\mathrm{F}) \leq \mathrm{a}$, and*

$$\mathrm{opt}_{\mathrm{E,weak}}(\mathrm{F}) \geq \begin{cases} ab/18 & \text{if } a \geq 2\log_2 b \\ 2^a - 1 & \text{if } a \leq \log_2 b \\ \frac{b}{3}\left(1 + \frac{1}{2}\ln\frac{2^a}{b}\right) & \text{otherwise.} \end{cases}$$

As a main application of Theorem 3.1 we obtain bounds on how much harder it is to learn with weak reinforcement.

Corollary 3.3 *If $F \subseteq Y^X$ then*

$$\mathrm{opt}_{\mathrm{E,weak}}(\mathrm{F}) \leq 1.39|\mathrm{Y}|\lceil 1 + \log_2 |\mathrm{Y}|\rceil \mathrm{opt}_{\mathrm{E,strong}}(\mathrm{F}).$$

Proof: Follows from (2) and the fact that a strong equivalence query can be simulated by a weak equivalence query and $\lceil \log_2 |Y| \rceil$ boolean queries. □

This bound is easily seen to be within an $O(\log |Y|)$ factor of the best possible.

Proof Sketch for Theorem 3.1: *The proof closely follows that of Theorem 2.1 (we therefore borrow notation from that proof). In the concept learning case we had the situation that a counterexample divides the subalgorithms into those which predicted correctly, respectively incorrectly, for the counterexample. Algorithm A^{E} (the master algorithm) achieved its performance by constructing its hypotheses in such a way that at least half of the weight predicted incorrectly for any counterexample.*

For strong reinforcement the same idea works if A^{E} constructs its hypothesis h such that $h(x)$ is a y for which $\sum_{i:h_i(x)=y} w_i$ is maximized over Y (with ties broken arbitrarily). When A^{E} receives a counterexample $(x^*, f(x^*))$, it is passed to all A_i^{EB} for which $h_i(x^*) \neq f(x^*)$ and all their weights are multiplied by 1/2. Since $h(x^*) \neq f(x^*)$ at most half of the weight predicted correctly for x^* and the analysis of the proof of Theorem 2.1 gives (1).

For weak reinforcement the situation is more complicated. Even if the hypotheses of A^{E} are constructed as in the strong reinforcement case, so that at most half of the weight predicted correctly for a counterexample x^*, the master algorithm A^{E} doesn't know which of the subalgorithms predicted incorrectly since it does not know the true value of $f(x^*)$. It only knows that those A_i^{EB} with $h_i(x^*) = h(x^*)$ predicted incorrectly. Therefore the counterexample x^* is passed only to these subalgorithms and only their weights w_i are multiplied by 1/2. Since $\sum_{i:h_i(x^*)=h(x^*)} w_i$ is at least a fraction $1/|Y|$ of the total weight before x^* was received, we can follow a line of reasoning analogous to that of Theorem 2.1 to see that the total weight decreases by at least a factor $1 - 1/(2|Y|)$ and thus $\mathrm{opt}_{\mathrm{E,weak}}(\mathrm{F}) \leq \mathrm{opt}_{\mathrm{EB,weak}}(\mathrm{F})/\log_2(1 + 1/(2|\mathrm{Y}| - 1))$, which gives (2).

To obtain a bound independent of $|Y|$ we consider a modified master algorithm, which also views the various copies of A^{EB} to be the leaves of a rooted tree with branching factor at most two in a manner similar to the example after the proof of Theorem 2.1. As in the algorithm A^{E} of Theorem 2.1, whenever a copy A_i^{EB} wants to ask a boolean query, its leaf is given two children corresponding to two copies generated and given YES *or* NO *respectively. However, as its hypothesis, the modified master algorithm chooses the hypothesis of some leaf A_i^{EB} of minimal depth in the tree. A child is given to this leaf which receives the counterexample x^*. The depth of the tree is bounded by $\mathrm{opt}_{\mathrm{EB,weak}}(\mathrm{F}) + 1$ and there always must be a leaf of depth at most $\mathrm{opt}_{\mathrm{EB,weak}}(\mathrm{F})$ corresponding to the "special" copy. The number of equivalence queries of A^{E} is given by the number of nodes in the tree with exactly 1 child. A combinatorial argument shows that this number is bounded by $2^{\mathrm{opt}_{\mathrm{EB,weak}}(\mathrm{F})} - 1$ which gives (3).* □

When $Y = \{0, 1\}$, a common way to build a richer class from a class $F \subseteq \{0,1\}^X$ is to take k-fold logical ORs of elements of F, i.e. $\{f_1 \vee \cdots \vee f_k : f_1, ..., f_k \in F\}$. Let us call the resulting class $\mathrm{OR}_k(F)$. Almost tight bounds on the PAC learning sample complexity of $\mathrm{OR}_k(F)$ in terms of k and the sample complexity of F have been obtained [6], but the techniques used apparently cannot be applied to bound the relative difficulty of *on-line* learning of $\mathrm{OR}_k(F)$, and this problem had remained open. Using our technique, we obtain the following result.

Theorem 3.4 *For any $F \subseteq \{0,1\}^X$, for any $k \geq 2$,*

$$\mathrm{opt}_{\mathrm{E}}(\mathrm{OR_k(F)}) \leq \frac{\mathrm{k}\lceil 1 + \log_2 \mathrm{k}\rceil \mathrm{opt_E(F)}}{\log_2 4/3}.$$

Proof: Choose X, $F \subseteq \{0,1\}^X$, and $k \geq 2$. For each $f_1, ..., f_k \in F$, let $\phi_{(f_1,...,f_k)} : X \to \{0,1\}^k$ be defined by $\phi_{(f_1,...,f_k)}(x) = (f_1(x), ..., f_k(x))$. Let

$$F_k = \{\phi_{(f_1,...,f_k)} : f_1, ..., f_k \in F\}.$$

To learn a $f_1 \vee \cdots \vee f_k \in \mathrm{OR}_k(F)$ it is sufficient to learn $\phi_{(f_1,...,f_k)} \in F_k$ in the weak reinforcement model since a hypothesis[4] $\phi_{(h_1,...,h_k)}$ for learning F_k can be transformed into a hypothesis $h_1 \vee \cdots \vee h_k$ for learning $\mathrm{OR}_k(F)$ and a counterexample x^* with $(h_1 \vee \cdots \vee h_k)(x^*) \neq (f_1 \vee \cdots \vee f_k)(x^*)$ also satisfies

[4] note that any function from X to $\{0,1\}^k$ can be expressed this way

$\phi_{(h_1,\ldots,h_k)}(x^*) \neq \phi_{(f_1,\ldots,f_k)}(x^*)$. Thus $\mathrm{opt_E}(\mathrm{OR_k(F)}) \leq \mathrm{opt_{E,weak}}(\mathrm{F_k})$ and similarly

$$\mathrm{opt_{EB}}(\mathrm{OR_k(F)}) \leq \mathrm{opt_{EB,weak}}(\mathrm{F_k}). \tag{4}$$

If the learner is allowed to ask boolean queries while learning F_k, after each counterexample x^*, $\lceil \log_2 k \rceil$ boolean queries suffice to find a component h_i of the hypothesis $\phi_{(h_1,\ldots,h_k)}$ for which $h_i(x^*) \neq f_i(x^*)$. Using copies of an optimal learning algorithm for F for each of the k components and passing the counterexample x^* to the corresponding copy of the incorrect component h_i implies

$$\mathrm{opt_{EB,weak}}(\mathrm{F_k}) \leq \lceil 1 + \log_2 \mathrm{k} \rceil \mathrm{k}\, \mathrm{opt_E}(\mathrm{F})$$

since the learner uses at most $k\,\mathrm{opt_E}(\mathrm{F})$ counterexamples and $\lceil \log_2 k \rceil$ times that many boolean queries.

Applying (4) and Theorem 2.1 completes the proof. □

This bound is easily seen to be within an $O(\log k)$ factor of the best possible. Also, it can be trivially generalized to functions with larger ranges, and to methods of combining elements of F other than taking ORs (see [3]).

One obtains a more realistic model of learning if the information given to the learner is sometimes incorrect. We might define $\mathrm{opt_E}(\mathrm{F}, \eta)$ to be the optimal number of equivalence queries required for learning F, given that at most η of the counterexamples were incorrect, i.e. were elements of the domain for which the learner's hypothesis was in fact correct. Clearly $\mathrm{opt_E}(\mathrm{F}, 0) = \mathrm{opt_E}(\mathrm{F})$.

If the learner is allowed to ask (using boolean queries) whether the last counterexample was incorrect or not, it can learn using only $\mathrm{opt_E}(\mathrm{F}, 0) + \eta$ equivalence queries and also that many boolean queries. This is achieved by simulating a learning algorithm with performance $\mathrm{opt_E}(\mathrm{F}, 0)$ in such a way that only correct counterexamples are passed to that algorithm. The proof of Theorem 2.1 thus gives the following result essentially without modification.

Theorem 3.5 *If $F \subseteq \{0,1\}^X$ then*

$$\mathrm{opt_E}(\mathrm{F}, \eta) \leq \frac{2(\mathrm{opt_E}(\mathrm{F}, 0) + \eta)}{\log_2 4/3} \leq 4.82(\mathrm{opt_E}(\mathrm{F}, 0) + \eta).$$

Cesa-Bianchi, et al [8] independently obtained a similar result which implies an improvement on Theorem 3.5 in which the constant is 4.41. Theorem 3.5 is optimal to within a constant factor since $\mathrm{opt_E}(\mathrm{F}, \eta) \geq 2\eta + \mathrm{opt_E}(\mathrm{F}, 0)$ [16]. Optimizing for the constant on the η term in Theorem 3.5 yields the following.

Theorem 3.6 *For any $F \subseteq \{0,1\}^X$ and any $0 < \epsilon < 1/3$*

$$\mathrm{opt_E}(\mathrm{F}, \eta) \leq (2+\epsilon)\eta + \left(\frac{17}{\epsilon} \ln \frac{1}{\epsilon}\right) \mathrm{opt_E}(\mathrm{F}, 0).$$

Proof Sketch: *The proof is similar to and less involved than the proof of Theorem 4.1 and therefore details are omitted here. The main idea is to use the following variant of the algorithm of the proof of Theorem 2.1. Instead of setting the weights of the newly generated copies corresponding to the answers YES and NO of a boolean query (in this case to the query "Was the last counterexample noisy?") each to half of the current weight w, the copies are given weights $c_1 w$ and $(1-c_1)w$, respectively. Furthermore the weights of the copies receiving a counterexample are multiplied by some c_2 instead of 1/2. Choosing appropriate c_1 and c_2 and carrying through with an analogous argument to that in the proof of Theorem 2.1 yields the theorem.* □

4 Randomized learning in the presence of malicious noise

In this section, we illustrate the application of our techniques to randomized learning algorithms, obtaining a result analogous to Theorem 3.6, except that the constant on the η term becomes 1 instead of $2+\epsilon$.

The model of this section is due to Maass [17]. For $F \subseteq \{0,1\}^X$, we assume that before the learning process starts, the environment fixes an $f \in F$, and a sequence $(x_1, y_1), \ldots, (x_m, y_m)$ for which $|\{t : y_t \neq f(x_t)\}| \leq \eta$, i.e. the number of noisy examples is at most η. Learning is still an on-line process, i.e. the learner asks its initial equivalence query h_1 without seeing any examples, then it receives (x_1, y_1), which may be a counterexample, or may be a supporting example, i.e. may have $h_1(x_1) = y_1$. Then, the learner uses (x_1, y_1) to (possibly randomly) generate a second hypothesis h_2, and so on. We denote by $\mathrm{opt_{ER}}(\mathrm{F}, \eta)$ the maximum (i.e., the worst-case), over $f \in F$ and $(x_1, y_1), \ldots, (x_m, y_m)$ subject to the constraints described above, of the expectation (over the algorithm's randomization) of the number of examples for which $h_t(x_t) \neq y_t$ made by the optimal algorithm in this model; i.e., $\mathrm{opt_{ER}}(\mathrm{F}, \eta)$ is the average number of "mistakes" made by the optimal algorithm for the most difficult target f and finite sequence of (x_t, y_t) pairs with $|\{t : y_t \neq f(x_t)\}| \leq \eta$ for it.

Theorem 4.1 *For any $F \subseteq \{0,1\}^X$, and any $\alpha, \beta > 0$ with $\alpha + \beta < 1$,*

$$\begin{aligned} \mathrm{opt_{ER}}(\mathrm{F}, \eta) &\leq \frac{\ln(1/\alpha)\mathrm{opt_E}(\mathrm{F}, 0) + \ln(1/\beta)\eta}{2\ln(\frac{2}{1+\alpha+\beta})} \\ &\leq \frac{2\ln(1/\alpha)\mathrm{opt_{ER}}(\mathrm{F}, 0) + \ln(1/\beta)\eta}{2\ln(\frac{2}{1+\alpha+\beta})}. \end{aligned}$$

Straightforward application of recent results on combining the predictions of experts [22, 16, 23, 9] yield bounds in terms of $|F|$ instead of $\mathrm{opt_{ER}}(\mathrm{F}, 0)$. Choosing appropriate α and β we get the following bounds.

Corollary 4.2 *If* $F \subseteq \{0,1\}^X$ *then*

$$\begin{aligned} \mathrm{opt}_{\mathrm{ER}}(\mathrm{F},\eta) &\leq 2.21(\mathrm{opt}_{\mathrm{E}}(\mathrm{F},0)+\eta) \\ \mathrm{opt}_{\mathrm{ER}}(\mathrm{F},\eta) &\leq 3.29(\mathrm{opt}_{\mathrm{ER}}(\mathrm{F},0)+\eta). \end{aligned}$$

If $\eta \geq 2\,\mathrm{opt}_{\mathrm{E}}(\mathrm{F},0)$

$$\mathrm{opt}_{\mathrm{ER}}(\mathrm{F},\eta) \leq \eta + 2\sqrt{\eta\,\mathrm{opt}_{\mathrm{E}}(\mathrm{F},0)}\left(\ln\frac{\eta}{\mathrm{opt}_{\mathrm{E}}(\mathrm{F},0)}+2\right).$$

and if $\eta \geq 4\,\mathrm{opt}_{\mathrm{ER}}(\mathrm{F},0)$

$$\mathrm{opt}_{\mathrm{ER}}(\mathrm{F},\eta) \leq \eta + 2\sqrt{2\eta\,\mathrm{opt}_{\mathrm{ER}}(\mathrm{F},0)}\left(\ln\frac{\eta}{2\,\mathrm{opt}_{\mathrm{ER}}(\mathrm{F},0)}+2\right).$$

Our results also hold in a variant of the above model where the environment need not choose the sequence $(x_1,y_1),\ldots,(x_m,y_m)$ in advance but only has to choose the example (x_t,y_t) before the learner proposes its hypothesis h_t, so that the example (x_t,y_t) may depend on the hypotheses $h_1,\ldots,h_{t-1}$.

Proof of Theorem 4.1: *We prove the first inequality. The second follows from the fact [17] that* $\mathrm{opt}_{\mathrm{E}}(\mathrm{F},0) \leq 2\mathrm{opt}_{\mathrm{ER}}(\mathrm{F},0)$.

The proof is like that of Theorem 2.1, except for the following differences.

1. *The master algorithm* A^{ER} *runs copies of an optimal learning algorithm* A^{E} *for the noise-free case.*
2. *The hidden information in this case is whether a given example* (x_t,y_t) *is noisy. Thus each copy of* A^{E} *which predicted incorrectly for the example* (x_t,y_t) *is split into two copies, one which assumes that the example was noisy, and another which assumes that the example was not noisy. Clearly the example is only passed to that copy of* A^{E} *which assumes that the example was not noisy.*
3. *When the master algorithm splits a subalgorithm into two copies, the weights of these copies are not reduced by the same factor: the weight of the copy that assumes the example was not noisy is* α *times the original weight, the weight of the copy that assumes the example was noisy is* β *times the original weight.*
4. *Instead of setting* $h(x)$ *to* 1 *deterministically if the total weight of copies evaluating to* 1 *is more than the total weight of those evaluating to* 0*, the master algorithm randomly sets* $h(x)$ *to* 1 *with probability* $P_{\alpha,\beta}(r)$ *given by*

$$\frac{\ln(1-r(1-\alpha-\beta))}{\ln(1-r(1-\alpha-\beta))+\ln(1-(1-r)(1-\alpha-\beta))},$$

where r *is the weight of the copies evaluating to* 1 *divided by the total weight of all copies.*[5] *(Observe that* $P_{\alpha,\beta}(1-r) = 1-P_{\alpha,\beta}(r)$*.)*

[5] Essentially the same "transfer function" was used in a different context in [22, 9].

Observe that the master algorithm splits all those subalgorithms which predicted incorrectly for the last example (x_t,y_t)*, without regard to whether its own prediction was correct or not.*

Let r_t *be the fraction of the weights on the subalgorithms that predict incorrectly on the* t*th example, and let* $p_t = P_{\alpha,\beta}(r_t)$ *be the probability that the master algorithm predicts incorrectly. To prove the theorem it is sufficient to bound* $\sum_{t=1}^{m} p_t$ *from above. Since, regardless of the algorithm's actual prediction, a fraction of* r_t *of the weights is modified we obtain that the total weight of all subalgorithms decreases by a factor* $1-r_t(1-\alpha-\beta)$*. Thus*

$$\alpha^{\mathrm{opt}_{\mathrm{E}}(\mathrm{F},0)}\beta^{\eta} \leq \prod_{t=1}^{m}[1-r_t(1-\alpha-\beta)]$$

because the weight of the special copy is at least $\alpha^{\mathrm{opt}_{\mathrm{E}}(\mathrm{F},0)}\beta^{\eta}$*. Some calculations, using the fact (see [9]) that* $\forall r, 2\ln(\frac{2}{1+\alpha+\beta})P_{\alpha,\beta}(r) \leq -\ln(1-r(1-\alpha-\beta))$*, complete the proof.* □

5 PAC learning with queries

In the PAC learning model [20] the learner has to, with high probability, give a good approximation to a target concept f from some concept class $F \subseteq \{0,1\}^X$. In the standard PAC-model the learner learns about the target concept only by random examples $(x_1,f(x_1)),(x_2,f(x_2)),\ldots$ independently drawn from some distribution $\mathcal{D}$ over X and labeled according to the target concept f. There are well known lower and upper bounds on the number of random examples necessary for learning: the upper bounds are $O\left(\mathrm{VCdim}(\mathrm{F})\epsilon^{-1}\log\epsilon^{-1}+\epsilon^{-1}\log\delta^{-1}\right)$ [6] and $O\left(\mathrm{VCdim}(\mathrm{F})\epsilon^{-1}\log\delta^{-1}\right)$ [12], and the lower bound [10] is $\Omega\left(\mathrm{VCdim}(\mathrm{F})\epsilon^{-1}+\epsilon^{-1}\log\delta^{-1}\right)$. In this section we prove a lower bound for a variant of the PAC learning model proposed by Turán [19], where the learner may request random examples, ask equivalence queries, and ask boolean queries. Then the learner has to output with high probability a good approximation of the target concept. The performance of the learner is measured by the number of random examples plus the number of equivalence queries plus the number of boolean queries. Turán [19] showed that the VC-dimension of the concept class (up to a constant factor) is a lower bound on the learning complexity in this model. Clearly this general bound cannot be improved by more than a constant factor since $\mathrm{VCdim}(\{0,1\}^{\mathrm{X}}) = |\mathrm{X}|$ and $\{0,1\}^X$ can be learned exactly using $|X|$ equivalence or boolean queries. In this section, we apply our technique to improve on the constant of Turán's bound, using a somewhat simpler proof. For previous results in the PAC learning model and variants thereof, see the references in [19].

Definition 5.1 *An algorithm A is a PAC-learning algorithm for $F \subseteq \{0,1\}^X$ with equivalence and boolean queries if for all $\epsilon, \delta > 0$, there exists some integer $m_A(\epsilon,\delta)$ such that, for all distributions $\mathcal{D}$ over X and all $f \in F$, if A is given ϵ and δ as input, after requesting a total number of random examples, equivalence queries, and boolean queries bounded by $m_A(\epsilon,\delta)$, A halts, and with probability at least $1-\delta$, outputs a final hypothesis $h \in \{0,1\}^X$ such that $\mathcal{D}[x \in X : f(x) \neq h(x)] \leq \epsilon$. The probability is taken over all requested random examples and the possible randomization of A.*

We denote the performance of an optimal learning algorithm for concept class F in this model by $\mathrm{opt}_{\mathrm{PEB}}(\mathrm{F},\epsilon,\delta)$. The main result of this section is the following theorem.

Theorem 5.2 *For all $F \subseteq \{0,1\}^X$ with $\mathrm{VCdim(F)} = \mathrm{d} < \infty$, all $0 \leq \epsilon \leq 1/24$, and all $0 \leq \delta \leq 1/24$, we have*

$$\mathrm{opt}_{\mathrm{PEB}}(\mathrm{F},\epsilon,\delta) \geq \mathrm{d}/10.$$

The proof of the theorem evolves through some lemmas, where the central lemma is Lemma 5.4.

Lemma 5.3 *If $\mathrm{VCdim(F)} = \mathrm{d} < \infty$ then $\mathrm{opt}_{\mathrm{PEB}}(\mathrm{F},\epsilon,\delta) \geq \mathrm{opt}_{\mathrm{PEB}}(\{0,1\}^{\mathrm{X}},\epsilon,\delta)$ for any X with $|X| = d$.*

Lemma 5.4 *Let $|X| = d$ and $F = \{0,1\}^X$. Then there is an algorithm which for any $0 \leq \delta \leq 1$ exactly learns any $f \in F$ with probability at least $1-\delta$, using only equivalence queries and random examples drawn from the uniform distribution on X, such that the number of random examples and equivalence queries is bounded by $\mathrm{opt}_{\mathrm{PEB}}(\mathrm{F},\epsilon,\delta)(1+\frac{1}{\log_2 4/3}) + \frac{\mathrm{d}\epsilon}{\log_2 4/3}$ for all $0 \leq \epsilon \leq 1$.*

Proof: The proof is similar to the proof of Theorem 2.1. Again we construct a learning algorithm A^{PE} which does not use boolean queries and runs as subalgorithms copies of an optimal learning algorithm (for F) A^{PEB} which does use boolean queries. To the proof of Theorem 2.1 we add mechanisms for dealing with random examples and the final hypotheses of the subalgorithms.

At the beginning A^{PE} requests a list L of $\mathrm{opt}_{\mathrm{PEB}}(\mathrm{F},\epsilon,\delta)$ random examples. If a copy A_i^{PEB} requests a random example then the next example from L not previously passed to A_i^{PEB} (or one of its predecessors) is passed to A_i^{PEB}. Thus different copies might receive the same random example, but from the point of view of a single copy it receives independently drawn examples. The weights of the copies are not changed if they request random examples.

If a copy A_i^{PEB} proposes a final hypothesis h_i, the master algorithm A^{PE} treats this hypothesis like a hypothesis of an equivalence query. It constructs its own hypothesis h (for an equivalence query) as the weighted majority vote of the hypotheses of the subalgorithms (some of which are final hypotheses, some of which are equivalence queries). If a final hypothesis h_i predicts correctly for the counterexample x^* (i.e. $h_i(x^*) \neq h(x^*)$) then the state of the copy A_i^{PEB} is not changed. If it predicts incorrectly ($h_i(x^*) = h(x^*)$) then A_i^{PEB}'s final hypothesis h_i is replaced by a new hypothesis h_i' which has $h_i'(x^*) \neq h(x^*)$ and is otherwise the same as h_i. The corresponding weight w_i is multiplied by $1/2$. Subalgorithms asking equivalence queries for which x^* is a counterexample are updated as in the proof of Theorem 2.1.

Following the analysis of the proof of Theorem 2.1 we have that after M equivalence queries of A^{PE} with incorrect hypotheses, the total sum of all weights is at most $(3/4)^M$. On the other hand with probability $1-\delta$ we have for the first final hypothesis h_s of the special copy that $\mathcal{D}[x : h_s(x) \neq f(x)] = |\{x : h_s(x) \neq f(x)\}|/d \leq \epsilon$, where f is the target concept. Thus the final copy must be modified at most $d\epsilon$ times to be completely correct. Hence with probability $1-\delta$ the weight of the special copy is at least $(1/2)^{\mathrm{opt}_{\mathrm{PEB}}(\mathrm{F},\epsilon,\delta)+\mathrm{d}\epsilon}$ since the number of equivalence and boolean queries of the special copy is bounded by $\mathrm{opt}_{\mathrm{PEB}}(\mathrm{F},\epsilon,\delta)$. By taking logarithms and solving for M, we get

$$M \leq \frac{1}{\log_2 4/3}(\mathrm{opt}_{\mathrm{PEB}}(\mathrm{F},\epsilon,\delta) + \mathrm{d}\epsilon)$$

which, adding the number of random examples, implies the lemma. □

The following lemma's proof uses ideas from [6, 19].

Lemma 5.5 *Choose $0 \leq \epsilon \leq 1$ and $0 \leq \delta \leq 1$. If $|X| = d$ and $F = \{0,1\}^X$ and a (randomized) learning algorithm A uses less than $d(1/2 - \epsilon - \delta + \epsilon\delta)$ equivalence queries and random examples drawn from the uniform distribution on X, then there is an $f \in F$ such that with probability at least δ, the error of A's final hypothesis is greater than ϵ.*

Proof Sketch: *Assume without loss of generality that $X = \{1,...,d\}$, and furthermore, that whenever A asks an equivalence query h, it receives the least counterexample to h; i.e., if f is the target, its counterexample is the least x such that $h(x) \neq f(x)$. Note that, with this convention for generating counterexamples, when A receives a counterexample x, in addition to discovering $f(x)$, it also can determine $f(z)$ for $z < x$, as it knows that $f(z) = h(z)$. On the other hand A does not learn anything about $f(z)$ for $z > x$.*

Suppose that the target function f is chosen uniformly at random from $\{0,1\}^X$, or that, equivalently, each $f(x)$ is independently set to 1 with probability $1/2$. If A "knows" $f(x)$ on k elements of the domain, then, since each $f(x)$ was chosen independently at random, the expected number of misclassifications of A's final hypoth-

esis, given this "knowledge", is $(d-k)/2$. Thus

$$E(\textit{number of misclassifications}) = d/2 - E(\textit{number of points "known"})/2. \quad (5)$$

To determine a bound on the expected number of "known" points, first, each random example increases the number of known points by at most 1. Now, consider a particular equivalence query h. Suppose $u_1, u_2, ..., u_l$ are the points that are "unknown" when h is asked. Since the value of the target function f was chosen independently of the other elements of the domain, h is correct on each u_i independently with probability $1/2$. Thus, the expected number of points "discovered" by this equivalence query is at most

$$\sum_{i=1}^{l} i2^{-i} + l2^{-l} \leq 2.$$

Thus for each random example or equivalence query, the expected number of points "discovered" is at most 2, and therefore, if a total of m equivalence queries and random examples are asked, the total expected number of points "discovered" is at most $2m$.

Plugging into (5), we get that the expected number of misclassifications is at least $d/2 - m$. The fact that for a randomly chosen f, the expected number of misclassifications is at least $d/2 - m$ implies that there is a particular f for which the expected number of misclassifications is at least $d/2 - m$. If the error of A's final hypothesis is no worse than ϵ with probability at least $1-\delta$, than the expected error is at most $\epsilon(1-\delta)+\delta$. Thus solving $(d/2-m)/d \leq \epsilon(1-\delta)+\delta$ for m completes the proof. □

Combining Lemmas 5.3, 5.4, and 5.5 together with some straightforward calculations proves Theorem 5.2.

Let $\mathrm{opt_{PB}}(\mathrm{F}, \epsilon, \delta)$ be the performance of an optimal learning algorithm which uses only random examples and boolean queries, and let $\mathrm{opt_P}(\mathrm{F}, \epsilon, \delta)$ be the performance of an optimal learning algorithm which only uses random examples. Then using similar techniques as above, together with by now standard Chernov-bound techniques from [6, 5], one can prove that

$$\mathrm{opt_P}(\mathrm{F}, \epsilon, \delta) = \mathrm{O}((\mathrm{opt_{PB}}(\mathrm{F}, \epsilon/2, \delta/2) + \log(1/\delta))/\epsilon). \quad (6)$$

If $\mathrm{opt_{PM}}$ is defined similarly where membership queries replace boolean queries, Eisenberg and Rivest [11] showed that for all F with a certain property, $\mathrm{opt_{PM}}(\mathrm{F}, \epsilon, \delta) = \Omega(\log(1/\delta)/\epsilon)$. PAC learning using *only* boolean queries was studied by Kulkarni, Mitter and Tsitsiklis [13].

6 Boolean queries and computationally efficient algorithms

To discuss issues of computational efficiency, it is useful to chose an encoding scheme for elements of X and elements of F (see, e.g. [2]). For each n and s, an encoding scheme gives rise to the set X_n of all elements of X whose encoding has length at most n and F_s, the corresponding set for F and s. We assume boolean queries take constant time. To ask an equivalence query h, the learning algorithm must output an algorithm for computing h, and is charged for the time required for evaluating h at the counterexample received for h. We say that an algorithm for F is efficient if the time required for learning any function $f \in F$ is bounded by a polynomial in the length of the encoding of the longest counterexample received and the encoding of f.

Bshouty, Goldman, Hancock and Matar [7], for many concrete classes F, proved tight bounds on the number of equivalence queries required by an algorithm that learns arbitrary functions in F using polynomially many membership queries. Their upper bounds were obtained using efficient algorithms. The following result provides a general bound on the usefulness of few boolean queries (and therefore, for example, few membership queries) for designing efficient algorithms.

Theorem 6.1 *Choose $X, F \subseteq \{0,1\}^X$ and an encoding scheme $\mathcal{E}$ for X and F. Then if there is an efficient algorithm A^{EB} for learning F (w.r.t. $\mathcal{E}$) and there is a constant c such that A^{EB} asks at most $c(\log n + \log s)$ boolean queries, where n is the length of the longest counterexample received and s is the length of the target function, then there is an efficient algorithm A^{E} for learning F (w.r.t. $\mathcal{E}$) that makes no boolean queries.*

Proof Sketch for Theorem 6.1 *We use a slightly modified version of the algorithm transformation of the proof of Theorem 2.1, in which copies of A^{EB} that have used more than $q = c(\log n + \log s)$ boolean queries are eliminated from consideration. The special copy is clearly never eliminated. We show that at any given time, there are at most 2^q copies under consideration. To see this, it is useful to view the various copies of A^{EB} as the leaves of a binary tree (different from the tree considered in the proof of Theorem 3.1). The tree is initialized with the single copy at the root. At any given time, when a copy asks a boolean query, it is given two children corresponding to the copies that receive YES and NO respectively. After A^{E} asks equivalence queries, the number of copies remains constant. Thus, we may consider those updates as simply relabelling the leaves of the tree. Since the tree under consideration always has depth at most q, it has at most 2^q leaves.* □

7 Acknowledgements

We thank Manfred Warmuth for posing the problems solved in Theorems 3.5 and 4.1. We thank György Turán for his comments on an earlier draft of this paper, including pointing out an error. We thank Peter Bartlett for asking us the question partially answered by (6).

References

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

[2] D. Angluin and M. Kharitonov. When won't membership queries help? In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 444–454, New Orleans, May 1991. ACM.

[3] P. Auer and P.M. Long. Structural results for on-line learning models with and without queries, 1993. Submitted.

[4] P. Auer, P.M. Long, W. Maass, and G.J. Woeginger. On the complexity of function learning. *The 1993 Workshop on Computational Learning Theory*, 1993.

[5] G. Benedek and A. Itai. Learnability with respect to fixed distributions. *Theoretical Computer Science* 86(2):377–389, 1991.

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

[7] N.H. Bshouty, S.A. Goldman, T.R. Hancock, and S. Matar. Asking questions to minimize errors. *The 1993 Workshop on Computational Learning Theory*, 1993.

[8] N. Cesa-Bianchi, Y. Freund, D. Helmbold, and M.K. Warmuth. On-line prediction and conversion strategies. *The 1993 IMA European conference on Computational Learning Theory*, 1993.

[9] N. Cesa-Bianchi, Y. Freund, D.P. Helmbold, D. Haussler, R.E. Schapire, and M.K. Warmuth. How to use expert advice. *Proceedings of the 25th ACM Symposium on the Theory of Computation*, 1993.

[10] A. Ehrenfeucht, D. Haussler, M. Kearns, and L.G. Valiant. A general lower bound on the number of examples needed for learning. *Information and Computation*, 82(3):247–251, 1989.

[11] B. Eisenberg and R.L. Rivest. On the sample complexity of PAC-learning using random and chosen examples. *The 1990 Workshop on Computational Learning Theory*, pages 154–162, 1990.

[12] D. Haussler, N. Littlestone, and M.K. Warmuth. Predicting $\{0,1\}$-functions on randomly drawn points. Technical Report UCSC-CRL-90-54, University of California Santa Cruz, Computer Research Laboratory, December 1990. To appear in Information and Computation.

[13] S.R. Kulkarni, S.K. Mitter, and J.N. Tsitsiklis. Active learning using arbitrary binary valued queries. *Machine Learning*, 11(1), 1993.

[14] N. Littlestone. Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[15] N. Littlestone. *Mistake Bounds and Logarithmic Linear-threshold Learning Algorithms.* PhD thesis, UC Santa Cruz, 1989.

[16] N. Littlestone and M.K. Warmuth. The weighted majority algorithm. Technical Report UCSC-CRL-91-28, UC Santa Cruz, October 1991. To appear, *Information and Computation.*

[17] W. Maass. On-line learning with an oblivious environment and the power of randomization. *The 1991 Workshop on Computational Learning Theory*, pages 167–175, 1991.

[18] W. Maass and G. Turán. Lower bound methods and separation results for on-line learning models. *Machine Learning*, 9:107–145, 1992.

[19] G. Turán. Lower bounds for PAC learning with queries. *The 1993 Workshop on Computational Learning Theory*, 1993.

[20] L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[21] V.N. Vapnik and A.Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[22] V. Vovk. Aggregating strategies. In *Proceedings of the 3nd Workshop on Computational Learning Theory*, pages 371–383. Morgan Kaufmann, 1990.

[23] V. Vovk. Universal forecasting algorithms. *Information and Computation*, 96(2):245–277, 1992.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'