Learning Binary Relations and Total Orders

(Extended Abstract)

Sally A. Goldman Ronald L. Rivest Robert E. Schapire

MIT Laboratory for Computer Science
Cambridge, Massachusetts 02139

Abstract. We study the problem of designing polynomial prediction algorithms for learning binary relations. We study these problems under an on-line model in which the instances are drawn by the learner, by a helpful teacher, by an adversary or according to a probability distribution on the instance space. We represent the relation as an $n \times m$ binary matrix, and present results for when the matrix is restricted to have at most k distinct row types, and when it is constrained by requiring that the predicate form a total order.

1 Introduction

Acquiring information about a predicate relating the elements of two sets is an important problem for many domains. For example, one may wish to learn a "has-part" predicate relating a set of animals to a set of attributes. We consider the problem of designing prediction algorithms to learn binary relations where the learner has little information about the predicate forming the relation. We represent the relation as an $n \times m$ binary matrix. (One could also model it by a bipartite graph.) The learner is repeatedly given an element from each set and asked to predict the corresponding matrix entry. In such a scenario a significant fraction of the matrix may be queried, and so the natural dimension measure of the problem is not the size of an instance, but rather the size nm of the instance space. As this problem demonstrates, there are interesting concept classes with polynomial sized instance spaces.

Such concept classes are uninteresting in Valiant's [15] probably approximately correct (PAC) model of learning. In this model, instances are chosen randomly from an arbitrary unknown probability distribution on the instance space. A concept class is PAC-learnable if the learner after seeing a polynomial number of instances can output a hypothesis that is correct on all but an arbitrarily small fraction of the instances with high probability. Since the learner sees most of the probability weight of the instance space when presented with a polynomial number of instances, these concept classes are trivially PAC-learnable. A goal of our research is to build a framework in which to study such problems.

To study learning algorithms for these concept classes we extend the basic mistake bound model [6, 7, 9] to one in which a helpful teacher or the learner selects the query sequence, in addition to considering when instances are chosen by an adversary or according to a probability distribution on the instance space. Previously, helpful teachers have been used to provide counterexamples to conjectured concepts [1], or to break up the concept into smaller sub-concepts [11]. In our framework, the teacher only selects the presentation order for the instances.

Since the learner can be forced to make nm mistakes when learning an arbitrary matrix, we consider restricting the matrix to have at most k distinct row types. (Two rows are of the same type if they agree in all columns.) We present an efficient algorithm making at most $km+(n-k)\lg k$ mistakes when the learner chooses the query sequence. When the adversary selects the query sequence, we present an efficient algorithm for $k = 2$ that makes at most $2m + n - 2$ mistakes, and for arbitrary k we present an efficient algorithm making at most $km + n\sqrt{(k-1)m}$ mistakes. We prove any algorithm makes at least $km+(n-k)\lfloor \lg k \rfloor$ when an adversary selects the query sequence. We use the existence of projective geometries to prove a lower bound of $\Omega(\min\{n\sqrt{m}, m\sqrt{n}\})$ for a large class of algorithms against an adversary. Finally, we describe a technique to simplify the analysis of expected mistake bounds when the query sequence is chosen at random, and use it to prove an $O(km + nk\sqrt{H})$ expected mistake bound for a simple algorithm. (Here H is the maximum Hamming distance between any two rows.)

In the second half of this paper we require the binary relation to form a total order. The halving algorithm [3, 9] yields a good mistake bound against any query sequence. (The halving algorithm predicts according to the majority of the feasible concepts, and thus each mistake halves the number of concepts to consider.) So a second goal of this research is to develop efficient implementations of the halving algorithm. We describe a technique that uses a fully polynomial randomized approximation scheme (fpras) to implement a randomized version of the halving algorithm. We apply this technique using a fpras due to Dyer, Frieze, and Kannan [5] and Matthews [10] for counting the number of extensions of a partial order to obtain a polynomial prediction algorithm for learning a total order that makes $n \lg n + o(n \lg n)$ mistakes with very high probability against an adversary selected query sequence. The probability of making "too many" mistakes is taken over the coin flips of the learning algorithm and does not depend on the query sequence. We contrast this result with a $\Theta(n \lg n)$ bound when the learner selects the query sequence, and a $n-1$ mistake bound when

This paper prepared with support from NSF grant DCR-8607494, ARO Grant DAAL03-86-K-0171, and a grant from the Siemens Corporation. Authors' network addresses: sally@theory.lcs.mit.edu, rivest@theory.lcs.mit.edu, rs@theory.lcs.mit.edu.

a teacher selects the query sequence. Finally, we discuss how the halving algorithm may be used to construct an efficient counting algorithm.

2 Learning Scenario and Mistake Bound Model

In this section we give formal definitions and then discuss the learning scenario used in this paper.

For each $n \geq 1$, let X_n denote a finite *learning domain* where n is a natural dimension measure for the domain. Let $X = \bigcup_{n \geq 1} X_n$, and $x \in X$ denote an *instance*. Given a learning domain X, for each $n \geq 1$, let $\mathcal{C}_n \subseteq 2^{X_n}$ be a family of *concepts* on X_n. Let $\mathcal{C} = \bigcup_{n \geq 1} \mathcal{C}_n$ denote a *concept class* over domain X. Given any concept $c \in \mathcal{C}_n$, we say that x in c is a *positive instance* of c, and x in $X_n - c$ is a *negative instance* of c. A *prediction algorithm* for $\mathcal{C}_n$ is an algorithm that runs under the following scenario. A *learning session* consists of a set of *trials*. In each trial, the algorithm predicts if $x \in X_n$ is a positive instance of the *target concept* $c \in \mathcal{C}_n$, and is then given feedback. If the prediction was incorrect, the algorithm has made a *mistake*. A learner is *consistent* if, on every trial, there is some concept in $\mathcal{C}_n$ that agrees with the learner's prediction, as well as with the labeled instances observed on the preceding trials.

The number of mistakes made by an algorithm depends on the sequence of instances presented to the learner. We extend the mistake bound model to include several methods for the selection of instances. The *query sequence* is some permutation $\pi = \{x_1, x_2, \ldots, x_{|X_n|}\}$ of X_n where x_t is the instance presented to the learner at the t^{th} trial. We call the agent selecting the query sequence the *instance selector*. We consider the following instance selectors:

- **Learner** – The learner chooses π. To select x_t, the learner may use time polynomial in n and all information obtained in the first $t-1$ trials.
- **Helpful Teacher** – A teacher who knows the target concept and wants to minimize the learner's mistakes chooses π. To select x_t, the teacher uses knowledge of the target concept, $x_1, \ldots, x_{t-1}$, and the learner's predictions on $x_1, \ldots, x_{t-1}$. To avoid allowing the learner and teacher to have a coordinated strategy, in this scenario we consider the worst case mistake bound over all consistent learners.
- **Adversary** – The adversary who selected the target concept chooses π. This adversary, who tries to maximize the learner's mistakes, knows the learner's algorithm and has unlimited computing power.
- **Random** – In this model, π is selected randomly according to a uniform probability distribution on the permutations of X_n.

We consider how a prediction algorithm's performance depends on the instance selector. Let $\mathcal{A}^Z(\mathcal{C})$ denote the set of prediction algorithms for learning concept class $\mathcal{C}$ with instance selector Z. For prediction algorithm $A \in \mathcal{A}^Z(\mathcal{C})$, we define the *mistake bound* $\mathrm{MB}(A, \mathcal{C}_n)$ to be the worst case number of mistakes made by A for any target concept in $\mathcal{C}_n$ under any query sequence provided by Z. (When Z = adversary, $\mathrm{MB}(A, \mathcal{C}_n) = M_A(\mathcal{C}_n)$ as defined by Littlestone [9].) We say that A is a *polynomial prediction algorithm* if A makes each prediction in time polynomial in n.

3 Learning a Binary Matrix

We now present our results for learning an $n \times m$ binary matrix M. Unless the matrix is constrained, the learner can be forced to make nm mistakes. There are many ways to reflect the "structure" in a binary relation. In this section we model the natural situation in which the matrix has at most k distinct row types, yet no other restrictions are placed on the predicate forming the relation. We say an entry i, j of the matrix (M_{ij}) is *known* if the learner was previously presented that entry. We assume without loss of generality that the learner is never asked to predict the value of a known entry. We say rows i and i' are *consistent* if $M_{ij} = M_{i'j}$ for all columns j in which both entries i, j and i', j are known.

To motivate our problem, suppose the learner is at a zoo that has n animals and m features are of interest (e.g. "has-wings"). The learner wants to accurately predict if a given animal has a given feature. We model this problem as that of learning an $n \times m$ binary matrix M whose n rows correspond to animals and m columns correspond to features. Having only k row types corresponds to there being only k types of animals. The learning session proceeds as follows: the instance selector selects an animal i and asks the learner to predict whether animal i has feature j. The learner predicts the value of M_{ij}, and is then told (or is allowed to observe) whether animal i has feature j.

Clearly, any learning algorithm makes at least km mistakes for some matrix, regardless of the query sequence. If computational efficiency is not a concern, for all query sequences the halving algorithm [3, 9] provides a $km + n \lg k$ mistake bound. We obtain this result by using a simple counting argument on the size of the concept class.

3.1 Learner Selected Query Sequence

In this section, we present an efficient algorithm for learning the matrix when the learner selects the query sequence.

Theorem 1 *There exists a polynomial prediction algorithm that achieves a $km + (n-k)\lg k$ mistake bound with a learner selected query sequence.*

Proof Sketch: The query sequence selected is that of predicting the entries of the matrix in row major order. The learner begins assuming that $k = 1$ and increments its estimate for k as needed. The algorithm predicts M_{ij} according to the majority of row types seen that are consistent with row i, and thus if a mistake is made, then at least half of the row types can be eliminated. At most m mistakes are made for the first row of each type. For the remaining $n-k$ rows, at most $\lg k$ mistakes are made before all row types are eliminated. ■

Note that this algorithm need not know k a priori. Furthermore, it obtains the same mistake bound even if an adversary tells the learner which row to examine, and in what order to predict the columns, provided the learner sees all of a row before going on to the next. However, this problem becomes much harder if the adversary can select the query sequence without restriction.

3.2 Adversary Presentation

In this section, we consider upper and lower bounds on learning algorithms for handling adversary selected query sequences. We begin by giving an algorithm for $k = 2$.

Theorem 2 *There exists an algorithm that makes at most $2m + n - 2$ mistakes against an adversary selected query sequence for $k = 2$.*

Proof Sketch: The algorithm uses a graph G whose vertices are the rows of the matrix and that initially has no edges. We place an edge $\overline{ij}$ in G when it is known that rows i and j are inconsistent. To predict M_{ij} the algorithm uses a row of the same color in the 2-coloring of G. The prediction algorithm is designed so that except for at most $2m - 1$ mistakes, every mistake adds an edge to G between two nodes previously assigned the same color. Thus, after at most $n - 1$ additional mistakes, there is a unique 2-coloring of G, and no more mistakes are made. ■

An interesting theoretical question is to find a linear mistake bound for constant $k \geq 3$ when provided with a k-colorability oracle. We define the related *matrix k-complexity* problem as follows: given an $n \times n$ binary matrix M that is partially known, decide if there is some matrix with at most k row types that is consistent with M. The matrix k-complexity problem can be shown to be $\mathcal{NP}$-complete by a reduction from graph k-colorability. (Proof omitted.)

We now present a simple algorithm designed to learn a matrix with arbitrary complexity k when an adversary selects the query sequence. We say that an algorithm A is a *row-filter algorithm* if A makes its prediction for M_{ij} strictly as a function of j and all entries in the set I of rows consistent with row i and defined in column j. That is, A's prediction is $f(I, j)$ where f is some (possibly probabilistic) function. Consider the following row-filter algorithm, *ConsMajorityPredict*, in which $f(I, j)$ computes the majority vote of the entries in column j of the rows in I. (Guess randomly in the case of a tie.) *ConsMajorityPredict* only takes time linear in the number of known entries of the matrix to make a prediction.

Theorem 3 *The algorithm ConsMajorityPredict makes at most $km + n\sqrt{(k-1)m}$ mistakes against an adversary selected query sequence.*

Proof Sketch: We use a potential argument. For all i, let $d(i)$ be the number of rows consistent with row i. We define the *potential* of a partially known matrix to be $\Phi = \sum_{i=1}^{n} d(i)$. We can prove that Φ is decreased by at most $\frac{k-1}{k}n^2$ during the learning session, and furthermore, Φ is non-increasing.

Next we prove that the r^{th} mistake made when predicting an entry in column j of some row i of type z decreases Φ by at least $2(r-1)$. The key observation is that since $r-1$ mistakes have occurred, at least $r-1$ entries are known in column j of rows of type z. So if a mistake occurs there must be at least $r-1$ entries in I (and thus consistent with row i) that differ in column j with row i. Thus after the r^{th} mistake in column j of row type z, Φ is decreased by at least $\sum_{x=1}^{r} 2(x-1) \geq (r-1)^2$.

By using the above observation, we can prove that after M mistakes have been made the total decrease in Φ is at least $km(\frac{M}{km} - 1)^2$. Thus, combining our bounds on the decrease of Φ it follows that

$$km\left(\frac{M}{km} - 1\right)^2 \leq \frac{k-1}{k}n^2.$$

This implies the desired bound on M. ■

We note that by using the simpler argument that each mistake decreases Φ by at least 2, we obtain a $km + \frac{k-1}{2k}n^2$ mistake bound for *any* row-filter algorithm. Also, Manfred Warmuth [16] has independently given an algorithm based on a majority voting scheme that achieves an $O(km + n\sqrt{m \lg k})$ mistake bound.*

We now focus on lower bounds.

x	x		x			
	x	x		x		
		x	x		x	
			x	x		x
x				x	x	
	x				x	x
x		x				x

Figure 1: A projective geometry for $p = 2$, $m = 7$.

Theorem 4 *Any algorithm makes at least $km + (n-k)\lfloor \lg k \rfloor$ mistakes for $m \geq \lfloor \lg k \rfloor$ when the adversary selects the query sequence.*

Proof Sketch: The adversary starts by presenting entries in the first $\lg k$ columns and replying that each prediction is incorrect. Each row can now be classified as one of k row types. Next the adversary presents the remaining columns for one row of each type, again replying that each prediction is incorrect. ■

By applying this result when $k = 2$ we see that the upper bound of Theorem 2 is tight. (If $m \leq \lfloor \lg k \rfloor$ then we get an mn lower bound for any query sequence.) Also, by modifying this technique we obtain an $\Omega(km + (n-k)\lg k)$ lower bound against all instance selectors.

Now we present a lower bound on the number of mistakes an adversary can force against any row-filter algorithm.

Theorem 5 *Let p be a prime integer and let $m = (p^2 + p + 1)$. Any row-filter algorithm for learning a $2n \times m$ matrix with $m \geq n$ and $k \geq 2$ makes at least $n(p+1) = \Omega(n\sqrt{m})$ mistakes when the adversary selects the query sequence.*

Proof Sketch: We assume that the adversary knows the learner's algorithm and has access to any random bits it uses. (We prove a similar lower bound on the expected mistake bound when the adversary cannot access the random bits.)

Our proof depends upon the existence of a projective geometry Γ on m points and lines [4]. That is, there exists a set of m points and a set of m lines such that each line contains exactly $p+1$ points and each point is at the intersection of exactly $p+1$ lines. Furthermore, any pair of lines intersects at exactly one point, and any two points define exactly one line. Figure 1 shows a matrix representation of such a geometry; an "x" in entry i, j indicates that point j is on line i. We use the first n rows of Γ.

The matrix M consists of two row types: the odd rows are filled with ones and the even rows with zeros. The adversary's query sequence maintains the condition that an entry i, j is not revealed unless line $\lceil i/2 \rceil$ of Γ contains point j. We prove that $I = I'$ for entries $2i, j$ and $2i-1, j$, and thus the learner makes the same prediction for both entries. Since the rows are of opposite parity, the learner is forced to make a mistake on one of these entries. ■

We use a similar argument to get an $\Omega(m\sqrt{n})$ bound for $m < n$. Combined with the lower bound of Theorem 4

*His algorithm is not a row-filter algorithm.

we obtain a $\Omega(km + (n-k)\lg k + \min\{n\sqrt{m}, m\sqrt{n}\})$ lower bound.

Given this lower bound, one may question the $2m+n-2$ upper-bound for $k=2$. However, the algorithm described is not a row-filter algorithm. Also compared to our results for the learner selected query sequence, it is clear that allowing the learner to select the query sequence is quite helpful.

3.3 Random Presentation

In this section we consider when the learner is presented at each step with one of the remaining entries of the matrix selected uniformly and independently at random. We present a prediction algorithm that makes $O(km+nk\sqrt{H})$ mistakes on average where H is the maximum Hamming distance between any two rows of the matrix. We note that when $H = \Omega(\frac{m}{k})$ the result of Theorem 3 supersedes this result. A key result of this section is a proof relating two different probabilistic models for analyzing the mistake bounds under a random presentation.

We now define the algorithm *RandomConsistentPredict*. The learner makes its prediction on M_{ij} as follows: let I be the set of rows in M consistent with row i and known in column j. Choose one row i' of I uniformly at random and guess $M_{i'j}$. (If I is empty, then make a random guess.)

Theorem 6 *Let H be the maximum Hamming distance between any two rows of M. The expected number of mistakes made by RandomConsistentPredict is $O(km+nk\sqrt{H})$.*

Proof Sketch: Let U_t be the probability that the prediction rule makes a mistake on the $(t+1)^{st}$ step. The expected number of mistakes is $\sum_{t=0}^{S-1} U_t$, where $S = mn$.

The condition that exactly t entries are known makes the computation of U_t rather messy since the probability of having seen some entry of the matrix is *not* independent of knowing the others. Instead, we compute the probability V_t of a mistake under the simpler assumption that each entry of the matrix has been seen with probability t/S, independent of the rest of the matrix.

We first prove that $\sum_{t=0}^{S-1} V_t = O(km + nk\sqrt{H})$. We note that by its definition

$$V_t = \frac{1}{S}\sum_{i,j} \Pr[\text{mistake} \mid i,j \text{ presented next}].$$

After a lengthy probability computation we obtain a formula upper bounding V_t. We then use integrals to upper bound the discrete sum to obtain the stated result.

Finally, we prove the main result of this section, namely, $\sum_{t=0}^{S-1} U_t = O\left(\sum_{t=0}^{S-1} V_t\right)$. We obtain this result by noting that

$$\sum_{t=0}^{S-1} V_t = \sum_{r=0}^{S-1} U_r \sum_{t=0}^{S-1} \binom{S}{r} \left(\frac{t}{S}\right)^r \left(1-\frac{t}{S}\right)^{S-r}.$$

and then using Stirling's approximation to show that the coefficients of the U_r's on the right are bounded below by a constant. ■

4 The Halving Algorithm and Approximate Counting

We now begin our discussion of learning a binary relation on a set in which the predicate is restricted to form a total order. First we give background information on the halving algorithm and approximate counting schemes.

We now review the halving algorithm [3, 9]. Let $\mathcal{V}$ denote the set of concepts in $\mathcal{C}_n$ that are consistent with all previous queries. Given an instance x in X_n, for each concept in $\mathcal{V}$ the halving algorithm computes the value for x and predicts according to the majority. Finally, all concepts in $\mathcal{V}$ that are inconsistent with the correct prediction are deleted. Littlestone [9] shows that this algorithm makes at most $\lg|\mathcal{C}_n|$ mistakes. Now suppose the prediction algorithm maintains a set $\mathcal{V}'$, the set of all concepts in $\mathcal{C}_n$ consistent with all *incorrectly* predicted instances. Littlestone [9] also proves that this *space-efficient halving algorithm* makes at most $\lg|\mathcal{C}_n|$ mistakes. So for any prediction algorithm A that only remembers its mistakes, the number of instances stored by A is bounded by $\mathrm{MB}(A, \mathcal{C}_n)$.

We define the *approximate halving algorithm* to be the following generalization of the halving algorithm. Given instance x in X_n the approximate halving algorithm predicts in agreement with at least $\varphi|\mathcal{V}|$ of the concepts in $\mathcal{V}$ for some constant $0 < \varphi < 1/2$.

Theorem 7 *The approximate halving algorithm makes at most $\log_{(1-\varphi)^{-1}}|\mathcal{C}_n|$ mistakes for concept class $\mathcal{C}$.*

Proof: Each time a mistake is made, the number of elements that remain in the concept space are reduced by a factor of φ. Thus after at most $\log_{(1-\varphi)^{-1}}|\mathcal{C}_n|$ mistakes there is one consistent concept left in $\mathcal{C}_n$. ■

We note that the above result holds for a space efficient version of the approximate halving algorithm.

We now introduce the notion of an approximate counting scheme for the problem of counting the number of elements in a finite set $\mathcal{S}$. Sometimes exact counting can be done in polynomial time; however, the counting problem is often $\#\mathcal{P}$-complete and thus assumed to be intractable. (For a discussion of $\#\mathcal{P}$ see Valiant's paper [14].) For many $\#\mathcal{P}$-complete problems good approximations are possible [8, 12, 13]. A *randomized approximation scheme*, R, for a counting problem satisfies the following condition for all $\epsilon, \delta > 0$:

$$\Pr\left[\frac{|\mathcal{S}|}{(1+\epsilon)} \le R(n,\epsilon,\delta) \le |\mathcal{S}|(1+\epsilon)\right] \ge 1-\delta$$

where $R(n,\epsilon,\delta)$ is R's estimate on input n, ϵ, and δ. Such a scheme is *fully polynomial* if it runs in time polynomial in $n, \frac{1}{\epsilon}, \lg\frac{1}{\delta}$. For a further discussion see Sinclair's thesis [12].

4.1 Learning a Total Order

In this section we show how to use a fpras to implement a randomized version of the approximate halving algorithm, and apply this result for the problem of learning a total order on a set of n elements. For this concept class $X_n = \{1,\ldots,n\} \times \{1,\ldots,n\}$. An instance (i,j) is in the target concept iff $i \le j$ in the corresponding total order.

Under the teacher selected query sequence we obtain an $n-1$ mistake bound. Using any standard sorting algorithm, we achieve a $\Theta(n\lg n)$ mistake bound when the learner selects the query sequence. With the adversary instance selector, this problem is like that of sorting when an adversary selects the order of the comparisons. We present a polynomial prediction algorithm making $n\lg n + o(n\lg n)$ mistakes with very high probability.

We begin by showing how to use an exact counting algorithm to implement the halving algorithm.

Lemma 1 *Given a polynomial algorithm R to exactly count the number of concepts in $\mathcal{C}_n$ consistent with a given set E of examples, one can construct an efficient implementation of the halving algorithm for $\mathcal{C}$.*

Proof Sketch: The learner implements the halving algorithm making calls to R to help choose its predictions. To make a prediction for $x \in X_n$ the learner calls R with x as a positive (negative) example and predicts with the option producing the larger count. ■

We modify this basic technique to use a fpras instead of the exact counting algorithm to obtain an efficient implementation of a randomized version of the approximate halving algorithm.

Theorem 8 *Let R be a* fpras *for counting the number of concepts in $\mathcal{C}_n$ consistent with a given set E of examples. If $|X_n|$ is polynomial in n, one can produce a prediction algorithm that for any $\delta > 0$ runs polynomial in n and $\lg \frac{1}{\delta}$ and makes $(1+o(1))\lg|\mathcal{C}_n|$ mistakes with probability at least $1-\delta$.*

Proof Sketch: The prediction algorithm implements the procedure described in Lemma 1 with the exact counting algorithm replaced by the fpras $R(n, \frac{1}{n}, \frac{\delta}{2|X_n|})$. Consider the prediction for x in X_n. Let $\mathcal{V}$ be the set of concepts that are consistent with all previous instances. Let r_1 (r_0) be the number of concepts in $\mathcal{V}$ for which x is a positive (negative) instance. Let $\hat{r}_1$ (respectively $\hat{r}_0$) be the estimate output by R for r_1 (r_0). Without loss of generality, assume the algorithm predicts that x is a negative instance, and thus $\hat{r}_0 \geq \hat{r}_1$. Applying the definition of a fpras and the observation that $r_0 + r_1 = |\mathcal{V}|$, gives that $r_0 \geq \frac{|\mathcal{V}|}{1+(1+\epsilon)^2}$.

We define an *appropriate* prediction to be a prediction that agrees with *at least* $\frac{|\mathcal{V}|}{1+(1+\epsilon)^2}$ of the concepts in $\mathcal{V}$. A straightforward argument yields that the probability all $|X_n|$ predictions are appropriate is at least $1-\delta$. When all predictions are appropriate this algorithm implements the approximate halving algorithm with $\varphi = \frac{1}{1+(1+\epsilon)^2}$ and thus, by Theorem 7, at most $\log_{(1-\varphi)^{-1}}|\mathcal{C}_n|$ mistakes are made. Substituting ϵ with its value of $\frac{1}{n}$ and simplifying the expression we obtain that with probability at least $1-\delta$, the number of mistakes is at most

$$\frac{\lg|\mathcal{C}_n|}{\lg\frac{1}{1-\varphi}} = (1+o(1))\lg|\mathcal{C}_n|.$$

■

We note that in the proof above, we could increase the probability of obtaining a $(1+o(1))\lg|\mathcal{C}_n|$ mistake bound since the desired mistake bound follows even if $o(\lg|\mathcal{C}_n|)$ of the predictions are not appropriate.

We now apply this result to obtain the main result of this section.

Theorem 9 *There exists an algorithm A_a in $\mathcal{A}^{\text{adversary}}(\mathcal{C})$ for learning total orders such that for all $\delta > 0$, A_a runs in time polynomial in n and $\lg\frac{1}{\delta}$ and makes at most $n\lg n + o(n\lg n)$ mistakes with probability at least $1-\delta$.*

Proof Sketch: We apply the results of Theorem 8 using the fpras for counting the number of extensions of a partial order given independently by Dyer, Frieze and Kannan [5], and Matthews [10]. We know that with probability at least $1-\delta$, the number of mistakes is $(1+o(1))\lg|\mathcal{C}_n|$. Since $|\mathcal{C}_n| = n!$ the desired result is obtained. ■

So for learning total orders for all instance selectors except for the helpful teacher, a polynomial prediction algorithm can do equally well asymptotically if a small probability of failure is allowed. However, for learning a binary matrix we have shown that when using a row-filter algorithm a learner can do asymptotically better under the learner instance selector versus the adversary instance selector.

Finally, we extend the mistake bound model to accommodate such randomized prediction algorithms. Although we are interested in worst case bounds, we now allow the algorithm to exceed the mistake bound with a specified probability δ that is taken over the coin flips of the prediction algorithm. We define the *randomized mistake bound (RMB)* as follows. For prediction algorithm $A \in \mathcal{A}^Z(\mathcal{C})$ with dimension measure n and confidence $\delta > 0$, we let $\text{RMB}(A, \mathcal{C}_n, \delta)$ denote a mistake bound m, such that with probability at least $1-\delta$, the worst case number of mistakes made by A is at most m. (The probability is taken over the coin flips used by A, and the mistake bound m is taken over all possible query sequences of Z.)

4.2 Majority Algorithms vs Counting Algorithms

In this section we discuss the relationship between counting schemes and the halving algorithm. Let $\mathcal{W}$ be a set of elements for which some subset $\mathcal{S}$ of the elements are distinguished. A *majority algorithm* takes as input (in some specified form) $\mathcal{V} \subseteq \mathcal{W}$ and outputs a bit that is 1 iff at least half of the elements in $\mathcal{V}$ are distinguished. On the other hand, a *counting algorithm* must output the number of elements in $\mathcal{V}$ that are distinguished.

In Lemma 1 we used a counting algorithm to implement a majority algorithm. We now discuss when a majority algorithm can be used to implement a counting algorithm.

Theorem 10 *Let A be a majority algorithm for $\mathcal{W}$ with running time T_A. Let $\mathcal{V}$ be any input to A, and suppose A outputs 1 (respectively, 0). Suppose further that it is possible in time $O(T_A)$ to create a new input $\mathcal{V}'$ (of the proper form) obtained by deleting $|\mathcal{V}|/2$ distinguished (non-distinguished) elements from $\mathcal{V}$. Then there exists an exact counting algorithm A' that runs in time $O(T_A \lg|\mathcal{V}|)$.*

We omit the proof here. We note that this result can be extended to apply if one can appropriately expand $\mathcal{V}$ to include $|\mathcal{V}|/2$ non-distinguished (distinguished) elements if A outputs 1 (respectively, 0). Applying Theorem 10 to k-CNF we obtain the following result.

Theorem 11 *There exists a polynomial time algorithm to exactly count the number of k-CNF formulas for which some $x \in X_n$ is a positive instance.*

Proof Sketch: Consider Valiant's [15] algorithm for k-CNF. As Angluin [2] notes, if Valiant's algorithm predicts 0, then there exists some clause τ that is 0. So by removing τ the condition needed for Theorem 10 is satisfied. ■

By studying the recursive structure of the counting algorithm, we obtain the following counting algorithm for k-CNF. Let T be the number of possible clauses that are true for instance x. Then the number of formulas that predict that x is a positive instance is 2^T. Also Theorem 11 can be generalized to prove that an approximate majority algorithm (i.e. the approximate halving algorithm) can be used to obtain an approximate counting algorithm. We omit the details here.

Concept Class	Instance Selector	Lower Bound	Upper Bound	Notes
$n \times m$ binary matrix (k row types)	Learner	km	$km + (n-k)\lg k$	
	Adversary	$km + (n-k)\lfloor \lg k \rfloor$	$O(km + n\sqrt{m \lg k})$ †	
	Adversary	$2m + n - 2$	$2m + n - 2$	$k = 2$
	Adversary	$\Omega(km + (n-k)\lg k + \min\{n\sqrt{m}, m\sqrt{n}\})$	$km + n\sqrt{(k-1)m}$	row-filter algorithm
	Uniform Dist.	km	$O(km + nk\sqrt{H})$	avg. case, row-filter alg.
Total Order on n elements	Teacher	$n - 1$	$n - 1$	
	Learner	$\Omega(n \lg n)$	$O(n \lg n)$	
	Adversary	$\Omega(n \lg n)$	$n \lg n + o(n \lg n)$	randomized mistake bound

Table 1: Summary of our results.

5 Conclusions and Open Problems

Our results are summarized in Table 1. All upper bounds are for polynomial prediction algorithms. The results listed are for deterministic mistake bounds unless otherwise stated. We have presented general techniques to help develop efficient versions of the halving algorithm. In particular, we have shown how a fpras can be used to efficiently implement a randomized version of the approximate halving algorithm. We have also extended the mistake bound model by adding the notion of an instance selector and generalizing it to accommodate randomized prediction algorithms.

In general, it would be nice to tighten any of the given bounds. We suspect that the projective geometry lower bound can be modified to incorporate k in the bound. Another nice problem is to find a "practical" prediction algorithm for the problem of learning a total order.

Acknowledgments

The results in Sections 4 and 4.1 were inspired by an "open problems session" led by Manfred Warmuth at our weekly Machine Learning Reading Group meeting, where he proposed the basic idea of using an approximate halving algorithm based on approximate and probabilistic counting, and also suggested the problem of learning a total order on n elements. We thank Tom Leighton for helping to improve the lower bound of Theorem 5. We also thank Nick Littlestone and Bob Sloan for their comments.

References

[1] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, November 1987.

[2] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

[3] J. Barzdin and R. Freivald. On the prediction of general recursive functions. *Sov. Math. Dokl.*, 13:1224–1228, 1972.

[4] R. Carmichael. *Introduction to the Theory of Groups of Finite Order*. Dover Publications, New York, 1937.

†Due to Manfred Warmuth.

[5] M. Dyer, A. Frieze, and R. Kannan. A random polynomial time algorithm for estimating the volumes of convex bodies. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 375–381, Seattle, Washington, May 1989.

[6] D. Haussler, M. Kearns, N. Littlestone, and M. Warmuth. Equivalence of models for polynomial learnability. In *First Workshop on Computational Learning Theory*, pages 42–55, Morgan-Kaufmann, August 1988.

[7] D. Haussler, N. Littlestone, and M. Warmuth. Expected mistake bounds for on-line learning algorithms. 1988. Unpublished manuscript.

[8] M. Jerrum and A. Sinclair. Conductance and the rapid mixing property for Markov chains: the approximation of the permanent resolved. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 235–244, May 1988.

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

[10] P. Matthews. Generating a random linear extension of a partial order. 1989. Unpublished manuscript.

[11] R. Rivest and R. Sloan. Learning complicated concepts reliably and usefully. In David Haussler and Leonard Pitt, editors, *First Workshop on Computational Learning Theory*, pages 69–79, Morgan Kaufmann, August 1988.

[12] A. Sinclair. *Randomised Algorithms for Counting and Generating Combinatorial Structures*. PhD thesis, University of Edinburgh, Department of Computer Science, November 1988.

[13] L. Stockmeyer. The complexity of approximate counting. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, pages 118–126, May 1983.

[14] L. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:198–201, 1979.

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

[16] M. Warmuth. Personal communication.