SIAM J. COMPUT.
Vol. 22, No. 5, pp. 1006–1034, October 1993

LEARNING BINARY RELATIONS AND TOTAL ORDERS*

SALLY A. GOLDMAN†, RONALD L. RIVEST‡, AND ROBERT E. SCHAPIRE§

Abstract. The problem of learning a binary relation between two sets of objects or between a set and itself is studied. This paper represents a binary relation between a set of size n and a set of size m as an $n \times m$ matrix of bits whose (i, j) entry is 1 if and only if the relation holds between the corresponding elements of the two sets. Polynomial prediction algorithms are presented for learning binary relations in an extended on-line learning model, where the examples are drawn by the learner, by a helpful teacher, by an adversary, or according to a uniform probability distribution on the instance space.

The first part of this paper presents results for the case in which the matrix of the relation has at most k row types. It presents upper and lower bounds on the number of prediction mistakes any prediction algorithm makes when learning such a matrix under the extended on-line learning model. Furthermore, it describes a technique that simplifies the proof of expected mistake bounds against a randomly chosen query sequence.

In the second part of this paper the problem of learning a binary relation that is a total order on a set is considered. A general technique using a fully polynomial randomized approximation scheme (fpras) to implement a randomized version of the halving algorithm is described. This technique is applied to the problem of learning a total order, through the use of an fpras for counting the number of extensions of a partial order, to obtain a polynomial prediction algorithm that with high probability makes at most $n \lg n + (\lg e)\lg n$ mistakes when an adversary selects the query sequence. The case in which a teacher or the learner selects the query sequence is also considered

Key words. machine learning, computational learning theory, on-line learning, mistake-bounded learning, binary relations, total orders, fully polynomial randomized approximation schemes

AMS subject classifications. 68Q25, 68T05

1. Introduction.

In many domains it is important to acquire information about a relation between two sets. For example, one may wish to learn a "has-part" relation between a set of animals and a set of attributes. We are motivated by the problem of designing a prediction algorithm to learn such a binary relation when the learner has limited prior information about the predicate forming the relation. Although one could model such problems as concept learning, they are fundamentally different problems. In concept learning there is a single set of objects and the learner's task is to classify these objects, whereas in learning a binary relation there are two sets of objects and the learner's task is to learn the predicate that relates the two sets. Observe that the problem of learning a binary relation can be viewed as a concept learning problem if one lets the instances be all ordered pairs of objects from the two sets. However, the ways in which the problem may be structured are quite different when the true task is to learn a binary relation as opposed to a classification rule. That is, instead of a rule that defines which objects belong to the target concept, the predicate defines a relationship between pairs of object.

*Received by the editors December 1, 1991; accepted for publication (in revised form) June 9, 1992. Most of this research was carried out while all three authors were at MIT Laboratory for Computer Science with support provided by National Science Foundation grant DCR-8607494, U.S. Army Research Office grant DAAL03-86-K-0171, Defense Advanced Research Projects Agency contract N00014-89-J-1988, and a grant from the Siemens Corporation. Preliminary versions of this paper appeared in the Proceedings of the 30th IEEE Symposium on Foundations of Computer Science, October 1989, and as Massachusetts Institute of Technology's Laboratory for Computer Science Technical Memo, MIT/LCS/TM-413, May 1990.

†Department of Computer Science, Washington University, St. Louis, Missouri 63130 (sg@cs.wustl.edu). This author currently receives support from G.E. Foundation Junior Faculty Grant and from National Science Foundation grant CCR-9110108.

‡MIT Laboratory for Computer Science, Cambridge, Massachusetts 02139 (rivest@theory.lcs.mit.edu).

§AT&T Bell Laboratories, Murray Hill, New Jersey 07974 (schapire@research.att.com). This author received additional support from U.S. Air Force Office of Scientific Research grant 89-0506 while at Harvard University.

A binary relation is defined between two sets of objects. Throughout this paper we assume that one set has cardinality n and the other has cardinality m. We also assume that for all possible pairings of objects the predicate relating the two sets of variables is either true (1) or false (0). Before defining a prediction algorithm, we first discuss our representation of a binary relation. Throughout this paper we represent the relation as an $n \times m$ binary matrix, where an entry contains the value of the predicate for the corresponding elements. Since the predicate is binary valued, all entries in this matrix are either 0 (false) or 1 (true). The *two-dimensional* structure arises from the fact that we are learning a binary relation.

For the sake of comparison we now briefly mention other possible representations. One could represent the relation as a table with two columns, where each entry in the first column is an item from the first set and each entry in the second column is an item from the second set. The rows of the table consist of the subset of the potential nm pairings for which the predicate is true. One could also represent the relation as a bipartite graph with n vertices in one vertex set and m vertices in the other set. An edge is placed between two vertices exactly when the predicate is true for corresponding items.

Having introduced our method for representing the problem, we now informally discuss the basic learning scenario. The learner is repeatedly given a pair of elements, one from each set, and is asked to predict the corresponding matrix entry. After making its prediction, the learner is told the correct value of the matrix entry. The learner wishes to minimize the number of incorrect predictions. Since we assume that the learner must eventually make a prediction for each matrix entry, the number of incorrect predictions depends on the size of the matrix.

Unlike problems typically studied, in which the natural measure of the size of the learner's problem is the size of an instance (or example), for this problem the natural measure is the size of the matrix. Such concept classes with polynomial-sized instance spaces are uninteresting in Valiant's probably approximately correct (PAC) model of learning [27]. 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 number of instances that are polynomial in the problem size, can output a hypothesis that is correct on all but an arbitrarily small fraction of the instances with high probability. For concepts whose instance space has cardinality polynomial in the problem size, by asking to see enough instances the learner can see almost all of the probability weight of the instance space. Thus it is not hard to show that these concept classes are trivially PAC-learnable. One goal of our research is to build a framework for studying such problems.

To study learning algorithms for these concept classes we extend the basic mistake bound model [14], [15], [19] to the cases in which a helpful teacher or the learner selects the query sequence, and, in addition, to the cases in which 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], [2] or to break up the concept into smaller subconcepts [23]. In our framework the teacher selects only the presentation order for the instances.

If the learner is to have any hope of doing better than random guessing, there must be some structure in the relation. Furthermore, since there are so many ways to structure a binary relation, we give the learner some prior knowledge about the nature of this structure. Not surprisingly, the learning task depends greatly on the prior knowledge provided. One way to impose structure is to restrict one set of objects to have relatively few types. For example, a circus may contain many animals but only a few different species. In the first part of this paper we study the case in which the learner has a priori knowledge that there are a limited number of object types. Namely, we restrict the matrix representing the relation to have at most k distinct row types. (Two rows are of the same type if they agree in all columns.) We

define a *k-binary-relation* to be a binary relation for which the corresponding matrix has at most k row types. This restriction is satisfied whenever there are only k types of objects in the set of n objects being considered in the relation. The learner receives *no* other knowledge about the predicate forming the relation. With this restriction we prove that any prediction algorithm makes at least $(1-\beta)km + n\lfloor \lg(\beta k)\rfloor - (1-\beta)k\lfloor \lg(\beta k)\rfloor$ mistakes[1] in the worst case for any fixed $0 < \beta \leq 1$ against any query sequence. So for $\beta = \frac{1}{2}$, we get a lower bound of $\frac{km}{2} + (n - \frac{k}{2})\lfloor \lg k - 1\rfloor$ on the number of mistakes made by any prediction algorithm. If computational efficiency is not a concern, the halving algorithm [4], [19] makes at most $km + (n-k)\lg k$ mistakes against any query sequence. (The halving algorithm predicts according to the majority of the feasible relations (or concepts), and thus each mistake halves the number of remaining relations.)

We present an efficient algorithm making at most $km + (n-k)\lfloor \lg k\rfloor$ mistakes in the case in which the learner chooses the query sequence. We prove a tight mistake bound[2] of $km + (n-k)(k-1)$ in the case in which the helpful teacher selects 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 that makes at most $km + n\sqrt{(k-1)m}$ mistakes. We prove that any algorithm makes at least $km + (n-k)\lfloor \lg k\rfloor$ mistakes in the case in which an adversary selects the query sequence, and we use the existence of projective geometries to improve this lower bound to $\Omega(km + (n-k)\lfloor \lg k\rfloor + \min\{n\sqrt{m}, m\sqrt{n}\})$ for a large class of algorithms. Finally, we describe a technique for simplifying the proof of expected mistake bounds when the query sequence is chosen at random, and we 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.)

Another possibility for known structure is the problem of learning a binary relation on a set where the predicate induces a total order on the set. (For example, the predicate may be "$<$".) In the second half of this paper we study the case in which the learner has *a priori* knowledge that the relation forms a total order. Once again, we see that the halving algorithm [4], [19] yields a good mistake bound against any query sequence. This motivates a second goal of this research: to develop efficient implementations of the halving algorithm. We uncover an interesting application of randomized approximation schemes to computational learning theory. Namely, 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 [10] and to Matthews [22] for counting the number of linear extensions of a partial order, to obtain a polynomial prediction algorithm that makes at most $n \lg n + (\lg e)\lg n$ mistakes with very high probability against an adversary-selected query sequence. The small probability of making too many mistakes is determined by the coin flips of the learning algorithm and not by the query sequence selected by the adversary. We contrast this result with an $n-1$ mistake bound when the learner selects the query sequence [28] and with an $n-1$ mistake bound when a teacher selects the query sequence.

The remainder of this paper is organized as follows. In the next section we formally introduce the basic problem, the learning scenario, and the extended mistake bound model. In §3 we present our results for learning k-binary-relations. We first give a motivating example and present some general mistake bounds. In the following subsections we consider query sequences selected by the learner, by a helpful teacher, by an adversary, or at random. In §4 we turn our attention to the problem of learning total orders. We begin by discussing

[1] Throughout this paper we use lg to denote $\log_2$.

[2] The tight mistake bound is a worst-case mistake bound taken over all consistent learners; see §2 for formal definitions.

the relationship between the halving algorithm and approximate counting schemes in §4.1. In particular, we describe how an fpras can be used to implement an approximate halving algorithm. Then in §4.2 we present our results on learning a total order. Finally, in §5 we conclude with a summary and discussion of related open problems.

2. Learning scenario and mistake bound model. In this section we give formal definitions and discuss the learning scenario used in this paper. To be consistent with the literature we discuss these models in terms of concept learning. As we have mentioned, the problem of learning a binary relation can be viewed in this framework by letting the instance space be all pairs of objects, one from each of the two sets.

A *concept* c is a Boolean function on some domain of instances. A *concept class* C is a family of concepts. The learner's goal is to infer some unknown target concept chosen from some *known* concept class. Often C is decomposed into subclasses C_n according to some natural dimension measure n. That is, for each $n \geq 1$ let X_n denote a finite *learning domain*. Let $X = \bigcup_{n\geq 1} X_n$, and let $x \in X$ denote an *instance*. To illustrate these definitions we consider the concept class of monomials. (A monomial is a conjunction of literals, where each literal is either some Boolean variable or its negation.) For this concept class n is just the number of variables. Thus $|X_n| = 2^n$, where each $x \in X_n$ is chosen from $\{0, 1\}^n$ and represents the assignment for each variable. For each $n \geq 1$ let C_n be a *family of concepts* on X_n. Let $C = \bigcup_{n\geq 1} C_n$ denote a *concept class* over X. For example, if C_n contains monomials over n variables, then C is the class of all monomials. Given any concept $c \in C_n$, we say that x is a *positive instance* of c if $c(x) = 1$, and we say that x is a *negative instance* of c if $c(x) = 0$. In our example the target concept for the class of monomials over five variables might be $x_1\overline{x_4}x_5$. Then the instance 10001 is a positive instance and the instance 00001 is a negative instance. Finally, the *hypothesis space* of algorithm A is simply the set of all hypotheses (or rules) h that A may output. (A hypothesis for C_n must make a prediction for each $x \in X_n$.)

A *prediction algorithm* for C is an algorithm that runs under the following scenario. A *learning session* consists of a set of *trials*. In each trial the learner is given an unlabeled instance $x \in X_n$. The learner uses its current hypothesis to predict whether x is a positive or negative instance of the target concept $c \in C_n$, and then the learner is told the correct classification of x. If the prediction is incorrect, the learner has made a *mistake*. Note that in this model there is no training phase. Instead, the learner receives *unlabeled instances* throughout the entire learning session. However, after each prediction the learner discovers the correct classification. This feedback can then be used by the learner to improve the learner's hypothesis. A learner is *consistent* if on every trial there is some concept in C_n that agrees both with the learner's prediction and with all the labeled instances observed on preceding trials.

The number of mistakes made by the learner depends on the sequence of instances presented. We extend the mistake bound model to include several methods for the selection of instances. A *query sequence* is a permutation $\pi = \langle x_1, x_2, \ldots, x_{|X_n|}\rangle$ of X_n, where x_t is the instance presented to the learner at the tth trial. We call the agent selecting the query sequence the *director*. We consider the following directors:

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. In this case we say that the learner is *self-directed*.

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. In this case we say the learner is *teacher directed*.

TABLE 1
Summary of testing reactions for allergy testing example.

Degree of patient allergy	Epicutaneous (scratch)	Intradermal (under the skin)
Not allergic	Negative	Negative
Mildly allergic	Negative	Weak positive
Highly allergic	Weak positive	Strong positive

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. In this case we say the learner is *adversary directed.*

Random. In this model, π is selected randomly according to a uniform probability distribution on the permutations of X_n. Here the number of mistakes made by the learner for some target concept c in C_n is defined to be the expected number of mistakes over all possible query sequences. In this case we say the learner is *randomly directed.*

We consider how a prediction algorithm's performance depends on the director. Namely, we let $\mathrm{MB}_Z(A, C_n)$ denote the worst-case number of mistakes made by A for any target concept in C_n when the query sequence is provided by Z. (When Z = adversary, $\mathrm{MB}_Z(A, C_n) = M_A(C_n)$ in the notation of Littlestone [19].) We say that A is a *polynomial prediction algorithm* if A makes each prediction in time polynomial in n.

3. Learning binary relations. In this section we apply the learning scenario of the extended mistake bound model to the concept class C of k-binary-relations. For this concept class the dimension measure is denoted by n and m and by $X_{n,m} = \{1, \ldots, n\} \times \{1, \ldots, m\}$. An instance (i, j) is in the target concept $c \in C_{n,m}$ if and only if the matrix entry in row i and column j is a 1. So in each trial the learner is repeatedly given an instance x from $X_{n,m}$ and is asked to predict the corresponding matrix entry. After making a prediction the learner is told the correct value of the matrix entry. The learner wishes to minimize the number of incorrect predictions during a learning session in which the learner must eventually make a prediction for each matrix entry.

We begin this section with a motivating example from the domain of allergy testing. We use this example to motivate both the restriction that the matrix has k row types and the use of the extended mistake bound model. We then present general upper and lower bounds on the number of mistakes made by the learner, regardless of the director. Finally, we study the complexity of learning a k-binary-relation under each director.

3.1. Motivation: Allergist example. In this subsection we use the following example taken from the domain of allergy testing to motivate the problem of learning a k-binary-relation.

Consider an allergist with a set of patients to be tested for a given set of allergens. Each patient is either highly allergic, mildly allergic, or not allergic to any given allergen. The allergist may use either an epicutaneous (scratch) test, in which the patient is given a fairly low dose of the allergen, or an intradermal (under the skin) test, in which the patient is given a larger dose of the allergen. The patient's reaction to the test is classified as strong positive, weak positive, or negative. Table 1 describes the reaction that occurs for each combination of allergy level and dosage level. Finally, we assume that a strong positive reaction is extremely uncomfortable to the patient but not dangerous.

What options does the allergist have in testing a patient for a given allergen? One option (option 0) is just to perform the intradermal test. Another option (option 1) is to perform an

epicutaneous test and, if it is not conclusive, then perform an intradermal test. (See Fig. 1 for decision trees describing these two testing options.) Which testing option is best? If the

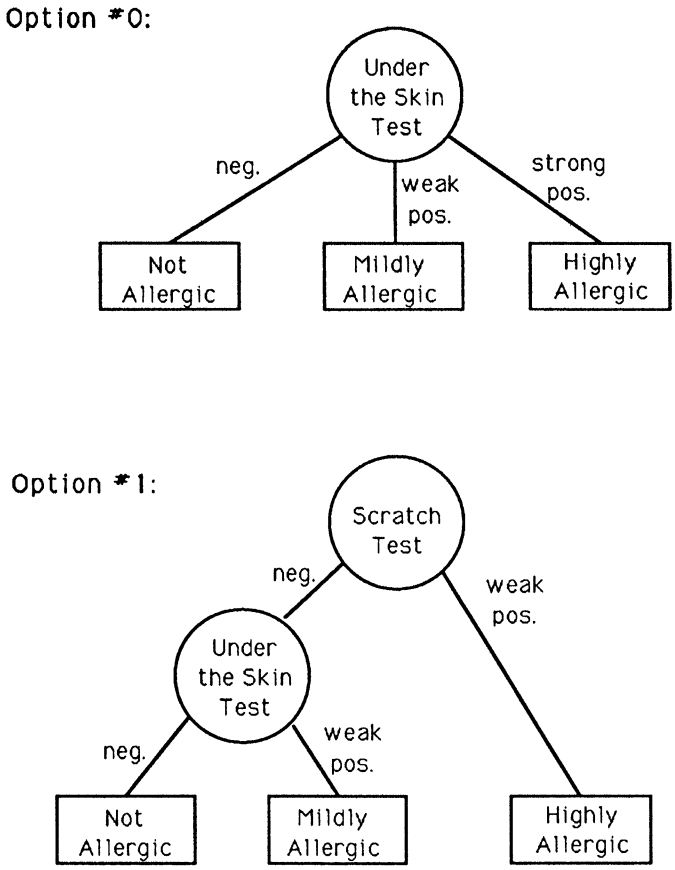


FIG. 1. *Testing options available to the allergist.*

patient has either no allergy or a mild allergy to the given allergen, then testing option 0 is best, since the patient need not return for the second test. However, if the patient is highly allergic to the given allergen, then testing option 1 is best, since the patient does not experience a bad reaction. We assume that the inconvenience of going to the allergist twice is approximately the same as having a bad reaction. That is, the allergist has no preference for an error in a particular direction. Although the allergist's final goal is to determine each patient's allergies, we consider the problem of learning the optimal testing option for each combination of patient and allergen.

The allergist interacts with the environment as follows. In each trial the allergist is asked to predict the best testing option for a given patient–allergen pair. The allergist is then told the testing results, thus learning whether the patient is not allergic, mildly allergic, or highly allergic to the given allergen. In other words, the allergist receives feedback as to the correct testing option. Note that we make no restrictions on how the hypothesis is represented, as long as it can be evaluated in polynomial time. In other words, all we require is that given any patient–allergen pair, the allergist decides which test to perform in a reasonable amount of time.

How can the allergist possibly predict a patient's allergies? If the allergies of the patients are completely random, then there is not much hope. What prior knowledge does the allergist have? He or she knows that people often have exactly the same allergies, and so there is a set of allergy types that occur frequently. (We do not assume that the allergist has *a priori* knowledge of the actual allergy types.) This knowledge can help guide the allergist's predictions.

Having specified the problem, we discuss our choice of using the extended mistake bound model to evaluate learning algorithms for this problem. First of all, observe that we want an on-line model. There is no training phase here; the allergist wants to predict the correct testing option for each patient–allergen pair. Also, we expect that the allergist has time to test each patient for each allergen; that is, the instance space is polynomial sized. Thus, as discussed in §1, the distribution-free model is not appropriate.

How should we judge the performance of the learning algorithm? For each wrong prediction made, a patient is inconvenienced by making a second trip or having a bad reaction. Since the learner wants to give all patients the best possible service, he or she strives to minimize the number of incorrect predictions made. Thus we want to use the absolute mistake bound success criterion. Namely, we judge the performance of the learning algorithm by the number of incorrect predictions made during a learning session in which the allergist must eventually test each patient for each allergen.

Up to now the standard on-line model (which uses absolute mistake bounds to judge the learners) appears to be the appropriate model. We now discuss the selection of the instances. Since the allergist has no control over the target relation (i.e., the allergies of the patients), it makes sense to view the feedback as coming from an adversary. However, do we really want an adversary to select the presentation order for the instances? It could be that the allergist is working for a cosmetic company and, because of the restrictions of the Food and Drug Administration and the cosmetic company, the allergist is essentially told when to test each person for each allergen. In this case it is appropriate to have an adversary select the presentation order. However, in the typical situation the allergist can decide in what order to perform the testing so that he or she can make the best predictions possible. In this case we want to allow the learner to select the presentation order. One could also imagine a situation in which an intern is being guided by an experienced allergist; in this case a teacher helps to select the presentation order. Finally, random selection of the presentation order may provide us with a better feeling for the behavior of an algorithm.

3.2. Learning k-binary-relations. In this section we begin our study of learning k-binary-relations by presenting general lower and upper bounds on the mistakes made by the learner, regardless of the director.

Throughout this section we use the following notation: 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 that rows i and i' are *consistent* (given the current state of knowledge) if $M_{ij} = M_{i'j}$ for all columns j in which both entries (i, j) and (i', j) are known.

We now look at general lower and upper bounds on the number of mistakes that apply for all directors. First of all, note that $k \leq 2^m$ since there are only 2^m possible row types for a matrix with m columns. Clearly, any learning algorithm makes at least km mistakes for some matrix, regardless of the query sequence. The adversary can divide the rows into k groups and reply that the prediction was incorrect for the first column queried for each entry of each group. We generalize this approach to force mistakes for more than one row of each type.

THEOREM 3.1. *For any* $0 < \beta \leq 1$ *any prediction algorithm makes at least* $(1 - \beta)km + n\lfloor \lg(\beta k) \rfloor - (1 - \beta)k\lfloor \lg(\beta k) \rfloor$ *mistakes, regardless of the query sequence.*

Proof. The adversary selects its feedback for the learner's predictions as follows. For each entry in the first $\lfloor \lg(\beta k) \rfloor$ columns the adversary replies that the learner's response is incorrect. At most βk new row types are created by this action. Likewise, for each entry in the first $(1-\beta)k$ rows the adversary replies that the learner's response is incorrect. This creates at most $(1-\beta)k$ new row types. The adversary makes all remaining entries in the matrix zero (see Fig. 2). The number of mistakes is at least the area of the unmarked region. Thus the adversary has forced at least $(1-\beta)km + n\lfloor \lg(\beta k) \rfloor - (1-\beta)k\lfloor \lg(\beta k) \rfloor$ mistakes while creating at most $\beta k + (1-\beta)k = k$ row types. □

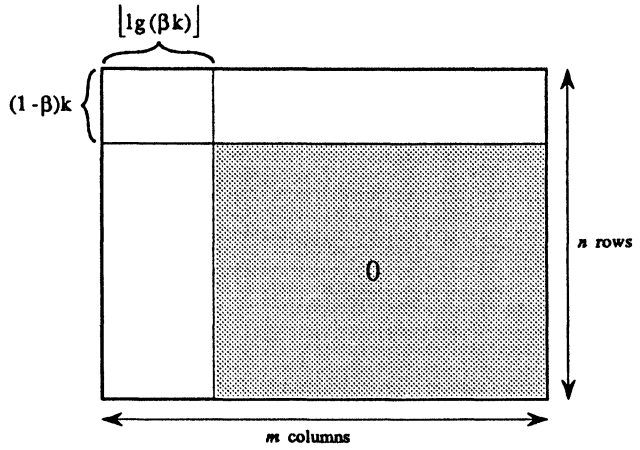


FIG. 2. *Final matrix created by the adversary in the proof of Theorem* 3.1. *All entries in the unmarked area will contain the bit not predicted by the learner; that is, a mistake is forced on each entry in the unmarked area. All entries in the marked area will be zero.*

By letting $\beta = \frac{1}{2}$ we obtain the following corollary.

COROLLARY 3.2. *Any algorithm makes at least* $\frac{km}{2} + (n - \frac{k}{2})\lfloor \lg k - 1 \rfloor$ *mistakes in the worst case, regardless of the query sequence.*

If computational efficiency is not a concern, for all query sequences the halving algorithm [4], [19] provides a good mistake bound.

Observation. The halving algorithm achieves a $km + (n-k)\lg k$ mistake bound.

Proof. We use a simple counting argument on the size of the concept class $C_{n,m}$. There are 2^{km} ways to select the k row types, and there are $k^{(n-k)}$ ways to assign one of the k row types to each of the remaining $n-k$ rows. Thus $|C_{n,m}| \leq 2^{km}k^{(n-k)}$. Littlestone [19] proves that the halving algorithm makes at most $\lg|C_{n,m}|$ mistakes. Thus the number of mistakes made by the halving algorithm for this concept class is at most $\lg(2^{km}k^{(n-k)}) \leq km + (n-k)\lg k$. □

In the remainder of this section we study efficient prediction algorithms designed to perform well against each of the directors. In some cases we are also able to prove lower bounds that are better than that of Theorem 3.1. In §3.3 we consider the case in which the query sequence is selected by the learner. We study the helpful-teacher director in §3.4. In §3.5 we consider the case of an adversary director. Finally, in §3.6 we consider instances drawn uniformly at random from the instance space.

3.3. Self-directed learning. In this section we present an efficient algorithm for the case of self-directed learning.

THEOREM 3.3. *There exists a polynomial prediction algorithm that achieves a* $km + (n - k)\lfloor \lg k \rfloor$ *mistake bound with a learner-selected query sequence.*

Proof. The query sequence selected simply specifies the entries of the matrix in row-major order. The learner begins by assuming that there is only one row type. Let $\hat{k}$ denote the learner's current estimate for k. Initially $\hat{k} = 1$. For the first row the learner guesses each entry. (This row becomes the template for the first row type.) Next the learner assumes that the second row is the same as the first row. If a mistake is made, then the learner revises the estimate for $\hat{k}$ to be 2, guesses for the rest of the row, and uses that row as the template for the second row type. In general, to predict M_{ij} the learner predicts according to a majority vote of the recorded row templates that are consistent with row i (breaking ties arbitrarily). Thus if a mistake is made, then at least half of the row types can be eliminated as the potential type of row i. If more than $\left\lfloor \lg \hat{k} \right\rfloor$ mistakes are made in a row, then a new row type has been found. In this case, $\hat{k}$ is incremented, the learner guesses for the rest of the row, and the learner makes this row the template for row type $\hat{k} + 1$.

How many mistakes are made by this algorithm? Clearly, at most m mistakes are made for the first row found of each of the k types. For the remaining $n - k$ rows, since $\hat{k} \leq k$, at most $\lfloor \lg k \rfloor$ mistakes are made. □

Observe that this upper bound is within a constant factor of the lower bound of Corollary 3.2. Furthermore, we note that this algorithm need not know k a priori. In fact, 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 that the learner sees all of a row before going on to the next. As we will later see, this problem becomes harder if the adversary can select the query sequence without restriction.

3.4. Teacher-directed learning. In this section we present upper and lower bounds on the number of mistakes made under the helpful-teacher director. Recall that in this model we consider the worst-case mistake bound over all consistent learners. Thus the question asked here is: What is the minimum number of matrix entries a teacher must reveal so that there is a unique completion of the matrix? That is, until there is a unique completion of the partial matrix, a mistake could be made on the next prediction.

We now prove an upper bound on the number of entries needed to uniquely define the target matrix.

THEOREM 3.4. *The number of mistakes made with a helpful teacher as the director is at most* $km + (n - k)(k - 1)$.

Proof. First the teacher presents the learner with one row of each type. For each of the remaining $n - k$ rows the teacher presents an entry to distinguish the given row from each of the $k - 1$ incorrect row types. After these $km + (n - k)(k - 1)$ entries have been presented we claim that there is a unique matrix with at most k row types that is consistent with the partial matrix. Since all k distinct row types have been revealed in the first stage, all remaining rows must be the same as one of the first k rows presented. However, each of the remaining rows have been shown to be inconsistent with all but one of these k row templates. □

Is Theorem 3.4 the best such result possible? Clearly, the teacher must present a row of each type. But, in general, is it really necessary to present $k - 1$ entries of the remaining rows to uniquely define the matrix? We now answer this question in the affirmative by presenting a matching lower bound.

THEOREM 3.5. *The number of mistakes made with a helpful teacher as the director is at least* $\min\{nm, km + (n - k)(k - 1)\}$.

Proof. The adversary selects the following matrix. The first row type consists of all zeros. For $2 \leq z \leq \min\{m + 1, k\}$, row type z contains $z - 2$ zeros, followed by a one, followed by

$m - z + 1$ zeros. The first k rows are each assigned to be a different one of the k row types. Each remaining row is assigned to be the first row type (see Fig. 3). Until there is a unique completion of the partial matrix, by definition there exists a consistent learner that could make a mistake. Clearly, if the learner has not seen each column of each row type, then the final matrix is not uniquely defined. This part of the argument accounts for km mistakes. When $m + 1 \geq k$, for the remaining rows, unless all of the first $k - 1$ columns are known, there is some row type besides the first row type that must be consistent with the given row. This argument accounts for $(n - k)(k - 1)$ mistakes. Likewise, when $m + 1 < k$, if any of the first m columns are not known then there is some row type besides the first row type that must be consistent with the given row. This accounts for $(n - k)m$ mistakes. Thus the total number of mistakes is at least $\min\{nm, km + (n - k)(k - 1)\}$. □

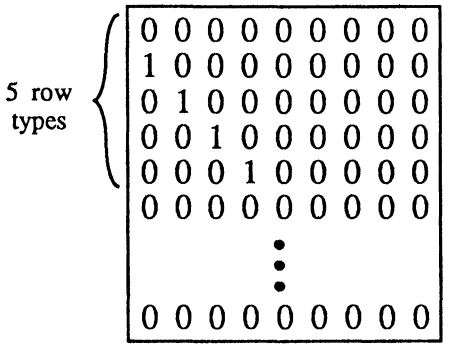


FIG. 3. *Matrix created by the adversary against the helpful teacher director. In this example five row types appear in the first five rows of the matrix.*

Because of the requirement that mistake bounds in the teacher-directed case apply to all consistent learners, we note that it is possible to get mistake bounds that are not as good as those obtained when the learner is self-directed. Recall that in §3.2 we proved a $km + (n - k)\lfloor \lg k \rfloor$ mistake bound for the learner director. This bound is better than that obtained with a teacher because the learner uses a majority vote among the known row types for making predictions. However, a consistent learner may use a *minority vote* and could thus make $km + (n - k)(k - 1)$ mistakes.

3.5. Adversary-directed learning. In this section we derive upper and lower bounds on the number of mistakes made when the adversary is the director. We first present a stronger information-theoretic lower bound on the number of mistakes an adversary can force the learner to make. Next we present an efficient prediction algorithm that achieves an optimal mistake bound if $k \leq 2$. We then consider the related problem of computing the minimum number of row types needed to complete a partially known matrix. Finally, we consider learning algorithms that work against an adversary for arbitrary k.

We now present an information-theoretic lower bound on the number of mistakes made by any prediction algorithm when the adversary selects the query sequence. We obtain this result by modifying the technique used in Theorem 3.1.

THEOREM 3.6. *Any prediction algorithm makes at least* $\min\{nm, km + (n - k)\lfloor \lg k \rfloor\}$ *mistakes against an adversary-selected query sequence.*

Proof. The adversary starts by presenting all entries in the first $\lfloor \lg k \rfloor$ columns (or m columns if $m < \lfloor \lg k \rfloor$) and by replying that each prediction is incorrect. If $m \geq \lfloor \lg k \rfloor$, this step causes the learner to make $n\lfloor \lg k \rfloor$ mistakes. Otherwise, this step causes the learner to

make nm mistakes. 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. For $m \geq \lfloor \lg k \rfloor$ this step causes the learner to make $k(m - \lfloor \lg k \rfloor)$ additional mistakes. For the remaining matrix entries the adversary replies as dictated by the completed row of the same row type as the given row. So the number of mistakes made by the learner is at least $\min\{nm, n\lfloor \lg k \rfloor + km - k\lfloor \lg k \rfloor\} = \min\{nm, km + (n-k)\lfloor \lg k \rfloor\}$. □

Special case: $k = 2$. We now consider efficient prediction algorithms for learning the matrix under an adversary-selected query sequence. (Recall that if efficiency is not a concern, the halving algorithm makes at most $km + (n-k)\lg k$ mistakes.) In this section we consider the case in which $k \leq 2$ and present an efficient prediction algorithm that performs optimally.

THEOREM 3.7. *There exists a polynomial prediction algorithm that makes at most* $2m + n - 2$ *mistakes against an adversary-selected query sequence for* $k = 2$.

Proof. The algorithm uses a graph G whose vertices correspond to the rows of the matrix and that initially has no edges. To predict M_{ij} the algorithm 2-colors the graph G and then proceeds as follows:

1. If no entry of column j is known, it guesses randomly.
2. Else if every known entry of column j is zero (respectively, one), it guesses zero (one).
3. Else it finds a row i' assigned the same color as i and known in column j, and it guesses $M_{i'j}$.

Finally, after the prediction is made and the feedback received, the graph G is updated by adding an edge $\overline{ii'}$ to G for each row i' known in column j for which $M_{ij} \neq M_{i'j}$. Note that one of the above cases always applies. Also, since $k = 2$, it will always be possible to find a 2-coloring.

How many mistakes can this algorithm make? It is not hard to see that cases 1 and 2 each occur only once for every column, and so there are at most m mistakes made in each of these cases. Furthermore, the first case-2 mistake adds at least one edge to G. We now argue that each case-3 mistake reduces the number of connected components of G by at least 1. We use a proof by contradiction. That is, assume that a case-3 mistake does *not* reduce the number of connected components. Then it follows that the edge $e = \overline{v_1 v_2}$ added to G must form a cycle (see Fig. 4). We now separately consider the cases in which this cycle contains an odd number of edges or an even number of edges.

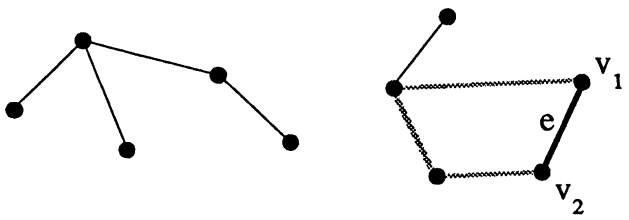


FIG. 4. *Situation that occurs if a case-3 mistake does not reduce the number of connected components of G. The thick gray edges and the thick black edge show the cycle created in G. Let e (shown as a thick black edge) be the edge added to form the cycle.*

Case 1: *Odd-length cycle.* Since G is known to be 2-colorable, this case cannot occur.

Case 2: *Even-length cycle.* Before e is added, since v_1 and v_2 were connected by an odd number of edges, in any legal 2-coloring they must have been different colors. Since step 3 of

the algorithm picks nodes of the same color, an edge could have never been placed between v_1 and v_2. Thus we again have a contradiction.

In both cases we reach a contradiction, and thus we have shown that every case-3 mistake reduces the number of connected components of G. Thus after at most $n-2$ case-3 mistakes, G must be fully connected and thus there must be a unique 2-coloring[3] of G and no more mistakes can occur. Thus the worst-case number of mistakes made by this algorithm is $2m+n-2$. □

Note that for $k=2$ this upper bound matches the information-theoretic lower bound of Theorem 3.6. Also note that if there is only one row type, then the algorithm given in Theorem 3.7 makes at most m mistakes, matching the information-theoretic lower bound.

An interesting theoretical problem is to find a linear mistake bound for constant $k \geq 3$ when provided with a k-colorability oracle. However, such an approach would have to be greatly modified to yield a polynomial prediction algorithm since a polynomial-time k-colorability oracle exists only if $\mathcal{P}=\mathcal{NP}$. Furthermore, even good polynomial-time approximations to a k-colorability oracle are not known [5], [18].

The remainder of this section focuses on designing polynomial prediction algorithms for the case in which the matrix has at least three row types. One approach that may seem promising is to make predictions as follows: Compute a matrix that is consistent with all known entries and that has the fewest possible row types; then use this matrix to make the next prediction. We now show that even computing the minimum number of row types needed to complete a partially known matrix is $\mathcal{NP}$-complete. Formally, we define the *matrix k-complexity* problem as follows: Given an $n \times m$ 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 for any fixed $k \geq 3$.

THEOREM 3.8. *For fixed $k \geq 3$ the matrix k-complexity problem is $\mathcal{NP}$-complete.*

Proof. Clearly, this problem is in $\mathcal{NP}$ since we can easily verify that a guessed matrix has k row types and is consistent with the given partial matrix.

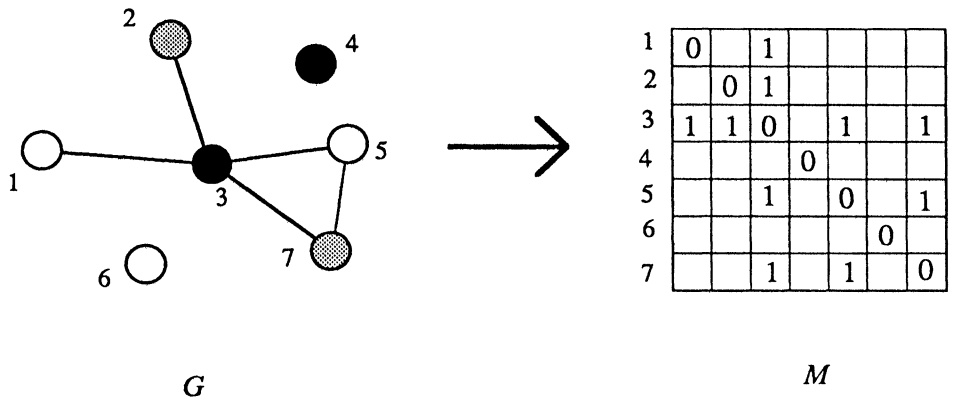
To show that the problem is $\mathcal{NP}$-complete, we use a reduction from graph k-colorability. Given an instance $G=(V, E)$ of graph k-colorability we transform it into an instance of the matrix k-complexity problem. Let $m=n=|V|$. For each edge $\{v_i, v_j\} \in E$ we add entries to the matrix so that row i and row j cannot be the same row type. Specifically, for each vertex v_i we set $M_{ii}=0$, and $M_{ji}=1$ for each neighbor v_j of v_i. An example demonstrating this reduction is given in Fig. 5.

We now show that there is some matrix of at most k row types that is consistent with this partial matrix if and only if G is k-colorable. We first argue that if there is a matrix M' consistent with M that has at most k row types, then G is k-colorable. By construction, if two rows are of the same type, there cannot be an edge between the corresponding nodes. So just let the node color for each node be the type of the corresponding row in M'.

Conversely, if G is k-colorable, then there exists a matrix M' consistent with M that has at most k row types. By the construction of M, if a set of vertices are the same color in G, then the corresponding rows are consistent with each other. Thus there exists a matrix with at most k row types that is consistent with M. □

Row-filter algorithms. In this section we study the performance of a whole class of algorithms 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

[3]Two 2-colorings under renaming of the colors are considered to be the same.



1	0		1				
2		0	1				
3	1	1	0		1		1
4				0			
5			1		0		1
6						0	
7			1		1		0

FIG. 5. *Example of the reduction used in Theorem* 3.8. *Graph G is the instance for the graph coloring problem, and partial matrix M is the instance for the matrix complexity problem. Note that there exists a matrix that is a completion of M that uses only three row types. The corresponding* 3*-coloring of G is demonstrated by the node colorings used in G.*

probabilistic) function. So to make a prediction for M_{ij} a row-filter algorithm considers all rows that could be the same type as row i and whose value for column j is known and uses these rows in any way one could imagine to make a prediction. For example, it could take a majority vote on the entries in column j of all rows that are consistent with row i. Or, of the rows defined in column j, it could select the row that has the most known values in common with row i and predict according to its entry in column j. We have found that many of the prediction algorithms we considered are row-filter algorithms.

Consider the simple 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.) Note that *ConsMajorityPredict* takes only time linear in the number of known entries of the matrix to make a prediction. We now give an upper bound on the number of mistakes made by *ConsMajorityPredict.*

THEOREM 3.9. *The algorithm ConsMajorityPredict makes at most* $km + n\sqrt{(k-1)m}$ *mistakes against an adversary-selected query sequence.*

Proof. 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 first consider how much the potential function can change over the entire learning session. □

LEMMA 3.10. *The potential function* Φ *decreases by at most* $\frac{k-1}{k}n^2$ *during the learning session.*

Proof. Initially, for all $i, d(i) = n$. So $\Phi_{\text{init}} = n^2$. Let $C(z)$ be the number of rows of type z for $1 \le z \le k$. By definition, $\Phi_{\text{final}} = \sum_{z=1}^{k} C(z)^2$. Thus our goal is to minimize $\sum_{z=1}^{k} C(z)^2$ under the constraint that $\sum_{z=1}^{k} C(z) = n$. Using the method of Lagrange multipliers, we obtain that Φ_{final} is minimized when for all z, $C(z) = n/k$. Thus $\Phi_{\text{final}} \ge (\frac{n}{k})^2 k = \frac{n^2}{k}$. So $\Delta\Phi = \Phi_{\text{init}} - \Phi_{\text{final}} \le n^2 - \frac{n^2}{k} = \frac{k-1}{k}n^2$. □

Now that the total decrease in Φ over the learning session is bounded, we need to determine how many mistakes can be made without Φ decreasing by more than $\frac{k-1}{k}n^2$. We begin by noting that Φ is strictly nonincreasing. Once two rows are found to be inconsistent, they remain inconsistent. So to bound the number of mistakes made by *ConsMajorityPredict* we must compute a lower bound on the amount Φ is decreased by each mistake. Intuitively, one expects Φ to decrease by larger amounts as more of the matrix is seen. We formalize this

intuition in the next two lemmas. For a given row type z let $B(j, z)$ denote the set of matrix entries that are in column j of a row of type z.

LEMMA 3.11. *The rth mistake made when predicting an entry in $B(j, z)$ causes Φ to decrease by at least $2(r-1)$.*

Proof. Suppose that this mistake occurs in predicting entry (i, j) where row i is of type z. Consider all the rows of type z. Since $r-1$ mistakes have occurred in column j, at least $r-1$ entries of $B(j, z)$ are known. Since *ConsMajorityPredict* is a row-filter algorithm, these rows must be in I. Furthermore, *ConsMajorityPredict* uses a majority voting scheme, and thus 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 if a mistake is made, row i is found to be inconsistent with at least $r-1$ rows it was thought to be consistent with. When two previously consistent rows are found to be inconsistent, Φ decreases by 2. Thus the total decrease in Φ caused by the rth mistake made when an entry is predicted in $B(j, z)$ is at least $2(r-1)$. □

From Lemma 3.11 we see that the more entries known in $B(j, z)$, the greater the decrease in Φ for future mistakes on such entries. So intuitively it appears that the adversary can maximize the number of mistakes made by the learner by balancing the number of entries seen in $B(j, z)$ for all j and z. We prove that this intuition is correct and apply it to obtain a lower bound on the amount Φ must have decreased after the learner has made μ mistakes.

LEMMA 3.12. *After μ mistakes are made, the total decrease in Φ is at least $km\left(\frac{\mu}{km}-1\right)^2$.*

Proof. From Lemma 3.11, after the rth mistake in the prediction of an entry from $B(j, z)$, the total decrease in Φ from its initial value is at least $\sum_{x=1}^{r} 2(x-1) \geq (r-1)^2$. Let $W(j, z)$ be the number of mistakes made in column j of rows of type z. The total decrease in Φ is at least

$$D = \sum_{j=1}^{m} \sum_{z=1}^{k} (W(j, z) - 1)^2,$$

subject to the constraint $\sum_{j=1}^{m} \sum_{z=1}^{k} W(j, z) = \mu$.

Using the method of Lagrange multipliers, we obtain that D is minimized when $W(j, z) = \frac{\mu}{km}$ for all j and z. (Since any algorithm clearly must make km mistakes, $\mu \geq km$ and thus $\mu/km \geq 1$.) So the total decrease in Φ is at least

$$\sum_{j=1}^{m} \sum_{z=1}^{k} \left(\frac{\mu}{km} - 1\right)^2 = km\left(\frac{\mu}{km} - 1\right)^2. \qquad \square$$

We now complete the proof of the theorem. Combining Lemma 3.10 and Lemma 3.12, along with the observation that Φ is strictly nonincreasing, we have shown that

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

This implies that $\mu \leq km + n\sqrt{(k-1)m}$. □

We note that by using the simpler argument that each mistake, except for the first mistake in each column of each row type, decreases Φ by at least 2, we obtain a $km + \frac{k-1}{2k} n^2$ mistake bound for *any* row-filter algorithm. Also, Goldman and Warmuth [12] give an algorithm, based on the weighted majority algorithm of Littlestone and Warmuth [20], that achieves an $O(km + n\sqrt{m \lg k})$ mistake bound. Their algorithm builds a complete graph of n vertices in which row i corresponds to vertex v_i and all edges have initial weights of 1. To predict a value for (i, j) the learner takes a weighted majority of all active neighbors of v_i (v_k is *active* if M_{kj} is known). After receiving feedback the learner sets the weight on the edge from v_i to v_k to

be 0 if $M_{kj} \neq M_{ij}$. Finally, if a mistake occurs, the learner doubles the weight of (v_i, v_k) if $M_{kj} = M_{ij}$ (i.e., the edges to neighbors that predicted correctly). We note that this algorithm is not a row-filter algorithm.

Does *ConsMajorityPredict* give the best performance possible by a row-filter algorithm? We now present an information-theoretic lower bound on the number of mistakes an adversary can force against any row-filter algorithm.

THEOREM 3.13. *Any row-filter algorithm for learning an $n \times m$ matrix with $m \geq \frac{n}{2}$ and $k \geq 2$ makes $\Omega(n\sqrt{m})$ mistakes when the adversary selects the query sequence.*

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

X	X		X			
	X	X		X		
		X	X		X	
			X	X		X
X				X	X	
	X				X	X
X		X				X

FIG. 6. *Projective geometry for $p = 2$, $m' = 7$.*

Let $m' = (p^2 + p + 1)$ be the largest integer of the given form such that p is prime and $m' \leq m$. Without loss of generality we assume in the remainder of this proof that the matrix has m' columns, and we prove an $\Omega(n\sqrt{m'})$ mistake bound. From Bertrand's conjecture[4] it follows from this result that the adversary has forced $\Omega(n\sqrt{m})$ mistakes in the original matrix.

Our proof depends on the existence of a *projective geometry* Γ on m' points and lines [6]. 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. (The choice of $m' = p^2 + p + 1$ for p prime comes from the fact that projective geometries are known to exist only for such values.) Figure 6 shows a matrix representation of such a geometry; an "X" in entry (i, j) indicates that point j is on line i. Let Γ' denote the first $\lfloor \frac{n}{2} \rfloor$ lines of Γ. Note that since $m' \geq \frac{n}{2}$, all entries of Γ' are contained within M.

The matrix M consists of two row types: The odd rows are filled with ones and the even rows with zeros. Two consecutive rows of M are assigned to each line of Γ' (see Fig. 7). We now prove that the adversary can force a mistake for each entry of Γ'. The adversary's query sequence maintains the condition that an entry (i, j) is not revealed unless line $\lceil \frac{i}{2} \rceil$ of Γ' contains point j. In particular, the adversary will begin by presenting one entry of the matrix for each entry of Γ'. We prove that for each entry of Γ' the learner must predict the same value for the two corresponding entries of the matrix. Thus the adversary forces a mistake for the $\lfloor \frac{n}{2} \rfloor (p+1) = \Omega(n\sqrt{m'})$ entries of Γ'. The remaining entries of the matrix are then presented in any order.

Let I be the set of rows that may be used by the row-filter algorithm when it predicts entry $(2i, j)$. Let I' be the set of rows that may be used by the row-filter algorithm when it predicts

[4] Bertrand's conjecture states that for any integer $n \geq 2$ there exists a prime p such that $n < p < 2n$. Although this is known as Bertrand's *conjecture*, it was proved by Chebyshev in 1831.

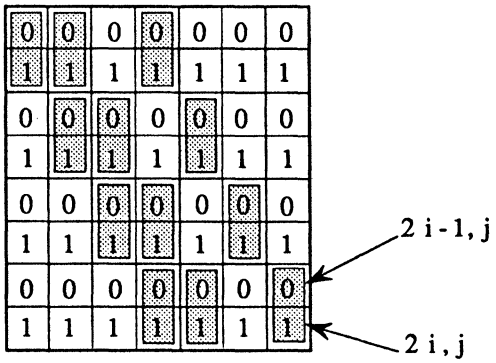


FIG. 7. *Matrix created by the adversary in the proof of Theorem* 3.13. *The shaded regions correspond to the entries in* Γ'. *The learner is forced to make a mistake on one of the entries in each shaded rectangle.*

entry $(2i-1, j)$. We prove by contradiction that $I = I'$. If $I \neq I'$, then it must be the case that there is some row r that is defined in column j and consistent with row $2i$, yet inconsistent with row $2i-1$ (or vice versa). By the definition of the adversary's query sequence it must be the case that lines $\lceil \frac{r}{2} \rceil$ and $\left\lceil \frac{(2i-1)}{2} \right\rceil = i$ of Γ' contain point j. Furthermore, since $(2i-1, j)$ is being queried, that entry is not known. Thus rows r and $2i-1$ must both be known in some other column j' since they are known to be inconsistent. Thus since only entries in Γ' are shown, it follows that lines $\lceil \frac{r}{2} \rceil$ and i of Γ' also contain point j' for $j' \neq j$. So this implies that lines $\lceil \frac{r}{2} \rceil$ and i of Γ' must intersect at two points, giving a contradiction. Thus $I = I'$, and so $f(I, j) = f(I', j)$ for entry $(2i, j)$ and entry $(2i-1, j)$. Since rows $2i$ and $2i-1$ differ in each column and the adversary has access to the random bits of the learner, the adversary can compute $f(I, j)$ just before making the query and then ask the learner to predict the entry for which the mistake will be made. This procedure is repeated for the pair of entries corresponding to each element of Γ'. □

We use a similar argument to get an $\Omega(m\sqrt{n})$ bound for $m < \frac{n}{2}$. This bound, combined with the lower bound of Theorem 3.6 and Theorem 3.13, permits us to obtain a $\Omega(km + (n-k)\lfloor \lg k \rfloor + \min\{n\sqrt{m}, m\sqrt{n}\})$ lower bound on the number of mistakes made by a row-filter algorithm.

COROLLARY 3.14. *Any row-filter algorithm makes* $\Omega(km+(n-k)\lfloor \lg k \rfloor + \min\{n\sqrt{m}, m\sqrt{n}\})$ *mistakes against an adversary-selected query sequence.*

Comparing this lower bound to the upper bound proven for *ConsMajorityPredict*, we see that for fixed k the mistake bound of *ConsMajorityPredict* is within a constant factor of optimal.

Given this lower bound, one may question the $2m+n-2$ upper bound for $k = 2$ given in Theorem 3.7. However, the algorithm described is not a row-filter algorithm. Also, compared to our results for the learner-selected query sequence, it appears that allowing the learner to select the query sequence is quite helpful.

3.6. Randomly directed learning. In this section we consider the case in which 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 3.9 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 first consider a simple probabilistic model in which the requirement that t matrix entries are known is simulated by assuming that each entry of the matrix is seen independently with probability $\frac{t}{nm}$. We then prove that any upper bound obtained on the number of mistakes under this simple probabilistic model holds under the true model (to within a constant factor) in which there are exactly t entries known. This result is extremely useful since in the true model the dependencies among the probabilities that matrix entries are known makes the analysis significantly more difficult.

We define the algorithm *RandomConsistentPredict* to be the row-filter algorithm in which the learner makes a prediction for M_{ij} by choosing one row i' of I uniformly at random and predicting the value $M_{i'j}$. (If I is empty, then *RandomConsistentPredict* makes a random guess.)

THEOREM 3.15. *Let H be the maximum Hamming distance between any two rows of M. Then the expected number of mistakes made by RandomConsistentPredict is $O(k(n\sqrt{H}+m))$.*

Proof. Let U_t be the probability that the prediction rule makes a mistake on the $(t+1)$st step. That is, U_t is the chance that a prediction error occurs on the next randomly selected entry, given that exactly t other randomly chosen entries are already known. Clearly, the expected number of mistakes is $\sum_{t=0}^{S-1} U_t$, where $S = nm$. Our goal is to find an upper bound for this sum.

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 $\frac{t}{S}$, independent of the rest of the matrix. We first compute an upper bound for the sum $\sum_{t=0}^{S-1} V_t$, and we then show that this sum is within a constant factor of $\sum_{t=0}^{S-1} U_t$.

LEMMA 3.16. $\sum_{t=0}^{S-1} V_t = O(km + nk\sqrt{H})$.

Proof. Fix t, and let $p = \frac{t}{S}$. Also, let $d(i)$ be the number of rows of the same type as row i. We bound V_0 by 1 trivially, and we assume henceforth that $p > 0$.

By definition, V_t is the probability of a mistake occurring when a randomly selected unknown entry is presented, given that all other entries are known with probability p. Since each entry (i, j) is presented next with probability $\frac{1}{S}$, it follows that

$$V_t = \frac{1}{S} \sum_{i,j} R_{ij},$$

where R_{ij} is the probability of a mistake occurring, given that entry (i, j) is unknown and is presented next.

Let I_{ij} be the random variable describing the set of rows consistent with row i and known in column j, and let J_{ij} be the random variable describing the set of rows i' in I_{ij} for which $M_{ij} \neq M_{i'j}$. If I_{ij} is nonempty, then the probability of choosing a row i' for which $M_{ij} \neq M_{i'j}$ is clearly $|J_{ij}| / |I_{ij}|$. Thus the probability of a mistake is just the expected value of this fraction, if it is assumed that $I_{ij} \neq \emptyset$.

Unfortunately, expectations of fractions are often hard to deal with. To handle this situation we therefore place a probabilistic lower bound on the denominator of this ratio, i.e., on $|I_{ij}|$. Note that if i and i' are of the same type, then the probability that $i' \in I_{ij}$ is just the chance p that (i', j) is known. Since there are $d(i)$ rows of type i (including i itself), we see that $\Pr[|I_{ij}| < y]$ is at most the chance that fewer than y of the other $d(i) - 1$ rows of the same type as i are in I_{ij}. In other words, this probability is bounded by the chance of fewer than y successes in a sequence of $d(i) - 1$ Bernoulli trials, each succeeding with probability p.

We use the following form of Chernoff bounds, due to Angluin and Valiant [3], to bound this probability:

LEMMA 3.17. *Consider a sequence of m independent Bernoulli trials, each succeeding with probability p. Let S be the random variable describing the total number of successes. Then for $0 \leq \gamma \leq 1$ the following hold:*

$$\Pr[S < (1-\gamma)mp] \leq e^{-\gamma^2 mp/2},$$

and

$$\Pr[S > (1+\gamma)mp] \leq e^{-\gamma^2 mp/3}.$$

Thus by letting $y = p(d(i)-1)/2$ and applying this lemma, it follows that

$$\Pr[|I_{ij}| < p(d(i)-1)/2] \leq e^{-p(d(i)-1)/8}.$$

Note that this bound applies even if $d(i) = 1$.

Thus we have

$$\begin{aligned} R_{ij} &\leq \Pr[|I_{ij}| < y] + \mathrm{E}\left[\frac{|J_{ij}|}{|I_{ij}|} \mid |I_{ij}| \geq y\right] \cdot \Pr[|I_{ij}| > y] \\ &\leq \Pr[|I_{ij}| < y] + \frac{\mathrm{E}[|J_{ij}| \mid |I_{ij}| \geq y]}{y} \cdot \Pr[|I_{ij}| \geq y] \\ &\leq \Pr[|I_{ij}| < y] + \frac{\mathrm{E}[|J_{ij}|]}{y}. \end{aligned}$$

So to bound R_{ij} it will be useful to bound $\mathrm{E}[|J_{ij}|]$.

We have

$$\mathrm{E}[|J_{ij}|] = \sum_{i' \neq i, M_{i'j} \neq M_{ij}} \Pr[i' \in I_{ij}].$$

If $M_{ij} \neq M_{i'j}$, then $\Pr[i' \in I_{ij}]$ is the chance that (i', j) is known and that i and i' are consistent. Entry (i', j) is known with probability p, and i and i' are consistent if either (i, j') or (i', j') is unknown for each column $j' \neq j$ in which i and i' differ. If $h(i, i')$ is the Hamming distance between rows i and i', then this probability is $(1-p^2)^{h(i,i')-1}$.

Combining these facts, we have

$$\begin{aligned} V_t &\leq \frac{1}{S}\sum_i \sum_j e^{-p(d(i)-1)/8} + \frac{1}{S}\sum_{d(i)>1} \sum_j \frac{\sum_{i' \neq i, M_{i'j} \neq M_{ij}} p(1-p^2)^{h(i,i')-1}}{p(d(i)-1)/2} \\ &= \frac{1}{n}\sum_i e^{-p(d(i)-1)/8} + \frac{1}{S}\sum_{d(i)>1} \sum_{i' \neq i} \frac{2h(i,i')(1-p^2)^{h(i,i')-1}}{d(i)-1}. \end{aligned}$$

Recall that our goal is to upper bound the sum $\sum_{t=0}^{S-1} V_t$. Applying the above upper bound for V_t, we get

$$\begin{aligned} \sum_{t=0}^{S-1} V_t &\leq \sum_{t=0}^{S-1} \left(\frac{1}{n}\sum_i e^{-(t/S)(d(i)-1)/8}\right) \\ &\quad + \sum_{t=0}^{S-1} \left(\frac{1}{S}\sum_{d(i)>1}\sum_{i' \neq i} \frac{2h(i,i')}{d(i)-1}\left(1-\left(\frac{t}{S}\right)^2\right)^{h(i,i')-1}\right). \end{aligned} \tag{1}$$

We now bound the first part of the above expression. We begin by noting that

$$\sum_{t=0}^{S-1}\left(\frac{1}{n}\sum_{i=1}^{n} e^{-(t/S)(d(i)-1)/8}\right) \le \frac{1}{n}\sum_{i=1}^{n}\left(1+\int_0^S e^{-(t/S)(d(i)-1)/8}dt\right)$$

$$\le \frac{1}{n}\sum_{i=1}^{n}\left(1+\frac{16S}{d(i)}\right),$$

where this last bound follows by evaluating the integral in the two cases that $d(i) = 1$ and $d(i) > 1$. This last expression equals

$$1 + 16m\sum_{i=1}^{n}\frac{1}{d(i)} = 16km + 1,$$

where the last step is obtained by rewriting the summation to go over all the row types: There are $d(i)$ terms for rows of the same type as row i; thus each row type contributes 1 to the summation.

We next bound the second part of expression (1). To complete the proof of the lemma it suffices to show that

$$\sum_{t=0}^{S-1}\left(\frac{1}{S}\sum_{d(i)>1}\sum_{i'\neq i}\frac{h(i,i')}{d(i)-1}\left(1-\left(\frac{t}{S}\right)^2\right)^{h(i,i')-1}\right) = O(nk\sqrt{H}).$$

We begin by noting that this expression is bounded above by

$$\frac{1}{S}\sum_{d(i)>1}\sum_{i'\neq i}\frac{h(i,i')}{d(i)-1}\left(1+\int_0^S\left(1-\left(\frac{t}{S}\right)^2\right)^{h(i,i')-1}dt\right).$$

If $h(i,i') = 1$, then this integral is trivially evaluated to be S. Otherwise, by applying the inequality $e^x \ge 1 + x$ we get

$$\int_0^S\left(1-\left(\frac{t}{S}\right)^2\right)^{h(i,i')-1}dt \le \int_0^S \exp\left\{-\left(\frac{t}{S}\right)^2(h(i,i')-1)\right\}dt. \tag{2}$$

A standard integral table [13] gives

$$\int_0^\infty \exp\left\{-\left(\frac{t}{S}\right)^2(h(i,i')-1)\right\}dt = \frac{S\sqrt{\pi}}{2\sqrt{h(i,i')-1}}. \tag{3}$$

Combining these bounds, we have

$$\int_0^S\left(1-\left(\frac{t}{S}\right)^2\right)^{h(i,i')-1}dt \le \frac{S\sqrt{\pi}}{\sqrt{2h(i,i')}} \tag{4}$$

for $h(i,i') \ge 1$. Thus we arrive at an upper bound of

$$\frac{1}{S}\sum_{d(i)>1}\sum_{i'\neq i}\frac{h(i,i')}{d(i)-1}\left(1+\int_0^S\left(1-\left(\frac{t}{S}\right)^2\right)^{h(i,i')-1}dt\right)$$

$$\leq \frac{1}{S}\sum_i \sum_{i' \neq i} \frac{2h(i,i')}{d(i)}\left(1 + \frac{S\sqrt{\pi}}{\sqrt{2h(i,i')}}\right)$$
$$\leq \frac{2m}{S}\sum_i \sum_{i' \neq i} \frac{1}{d(i)} + \sqrt{2\pi}\sum_i \sum_{i' \neq i} \frac{\sqrt{h(i,i')}}{d(i)}$$
$$= O(nk\sqrt{H}).$$

This implies the desired bound. □

To complete the theorem we prove the main result of this section, namely, that the upper bound obtained under this simple probabilistic model holds (to within a constant factor) for the true model. In other words, to compute an upper bound on the number of mistakes made by a prediction algorithm when the instances are selected according to a uniform distribution on the instance space, one can replace the requirement that exactly t matrix entries are known by the requirement that each matrix entry is known with probability $\frac{t}{nm}$.

LEMMA 3.18. $\sum_{t=0}^{S-1} U_t = O\left(\sum_{t=0}^{S-1} V_t\right)$.

Proof. We first note that

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

To see this, observe that for each r, where r is the number of known entries, we need only multiply U_r by the probability that exactly r entries are known if it is assumed that each entry is known with probability of $\frac{t}{S}$. Therefore,

$$\sum_{t=0}^{S-1} V_t = \sum_{t=0}^{S-1}\sum_{r=0}^{S-1} U_r \binom{S}{r}\left(\frac{t}{S}\right)^r \left(1 - \frac{t}{S}\right)^{S-r} \tag{5}$$
$$= \sum_{r=0}^{S-1} U_r \left[\sum_{t=0}^{S-1} \binom{S}{r}\left(\frac{t}{S}\right)^r \left(1 - \frac{t}{S}\right)^{S-r}\right].$$

Thus to prove the lemma it suffices to show that the inner summation is bounded below by a positive constant. By symmetry assume that $r \leq \frac{S}{2}$ and let $y = S - r$. Stirling's approximation implies that

$$\binom{S}{r} = \Theta\left(\frac{S^S}{r^r y^y}\sqrt{\frac{S}{ry}}\right).$$

Applying this formula to the desired summation, we obtain that

$$\sum_{t=0}^{S-1} \binom{S}{r}\left(\frac{t}{S}\right)^r \left(1 - \frac{t}{S}\right)^{S-r} = \Theta\left(\sqrt{\frac{S}{ry}}\sum_{t=0}^{S}\left(\frac{t}{r}\right)^r \left(\frac{S-t}{y}\right)^y\right)$$
$$= \Omega\left(\sqrt{\frac{S}{ry}}\sum_{x=1}^{\sqrt{ry/S}}\left(\frac{r+x}{r}\right)^r \left(\frac{y-x}{y}\right)^y\right). \tag{6}$$

The last step above follows by letting $x = t - r$ and reducing the limits of the summation. To complete the proof that equation (6) is bounded below by a positive constant we need prove

only that

$$\left(\frac{r+x}{r}\right)^r \left(\frac{y-x}{y}\right)^y = \Omega(1)$$

for all $1 \leq x \leq \sqrt{ry/S}$.

By using the inequality $1 + x \leq e^x$ it can be shown that for $1 + y > 0$, $1 + y \geq e^{y/(1+y)}$. We apply this observation to get that

$$\begin{aligned}
\left(\frac{r+x}{r}\right)^r \left(\frac{y-x}{y}\right)^y &= \left(1+\frac{x}{r}\right)^r \left(1-\frac{x}{y}\right)^y \\
&\geq \exp\left\{\frac{x}{1+\frac{x}{r}} - \frac{x}{1-\frac{x}{y}}\right\} \\
&= \exp\left\{\frac{rx}{r+x} - \frac{yx}{y-x}\right\} \\
&= \exp\left\{\frac{-x^2(r+y)}{(r+x)(y-x)}\right\} \\
&= \exp\left\{\frac{-Sx^2}{(r+x)(y-x)}\right\}.
\end{aligned}$$

Since $x \leq \sqrt{ry/S}$, it follows that $Sx^2 \leq ry$. By applying this observation to the above inequality it follows that

$$\begin{aligned}
\left(\frac{r+x}{r}\right)^r \left(\frac{y-x}{y}\right)^y &\geq \exp\left\{\frac{-ry}{(r+x)(y-x)}\right\} \\
&= \exp\left\{\frac{-ry}{ry+(y-r)x-x^2}\right\} \\
&\geq \exp\left\{\frac{-ry}{ry-\frac{ry}{S}}\right\} \\
&= \exp\left\{\frac{-1}{1-\frac{1}{S}}\right\}.
\end{aligned}$$

Finally, we note that for $S \geq 2$, $e^{-1/(1-1/S)} \geq e^{-2}$. This completes the proof of the lemma. □

Clearly, Lemma 3.16 and Lemma 3.18 together imply that $\sum_{t=0}^{S-1} U_t = O(km + nk\sqrt{H})$, giving the desired result. □

This completes our discussion of learning k-binary-relations.

4. Learning a total order. In this section we present our results for learning a binary relation on a set where it is known a priori that the relation forms a total order. One can view this problem as that of learning a total order on a set of n objects where an instance corresponds to comparing which of two objects is greater in the target total order. Thus this problem is like comparison-based sorting, except for two key differences: We vary the agent selecting the order in which comparisons are made (in sorting the learner does the selection), and we charge the learner only for *incorrectly predicted* comparisons.

Before describing our results, we motivate this section with the following example. There are n basketball teams that are competing in a round-robin tournament. That is, each team will play all other teams exactly once. Furthermore, we make the (admittedly simplistic) assumption that there is a ranking of the teams such that a team wins its match if and only if its opponent is ranked below it. A gambler wants to place a \$10 bet on each game: if he bets on the winning team he wins \$10 and if he bets on the losing team he loses \$10. Of course, his goal is to win as many bets as possible.

We formalize the problem of learning a total order as follows. The instance space $X_n = \{1, \ldots, n\} \times \{1, \ldots, n\}$. An instance (i, j) in X_n is in the target concept if and only if object i precedes object j in the corresponding total order.

If computation time is not a concern, then the halving algorithm makes at most $n \lg n$ mistakes. However, we are interested in efficient algorithms, and thus our goal is to design an efficient version of the halving algorithm. In the next section we discuss the relation between the halving algorithm and approximate counting. Then we show how to use an approximate counting scheme to implement a randomized version of the approximate halving algorithm, and we apply this result to the problem of learning a total order on a set of n elements. Finally, we discuss how a majority algorithm can be used to implement a counting algorithm.

4.1. The halving algorithm and approximate counting. In this section we review the halving algorithm and approximate counting schemes. We first cover the halving algorithm [4], [19]. Let $\mathcal{V}$ denote the set of concepts in C_n that are consistent with the feedback from all previous queries. Given an instance x in X_n, for each concept in $\mathcal{V}$ the halving algorithm computes the prediction of that concept for x and predicts according to the majority. Finally, all concepts in $\mathcal{V}$ that are inconsistent with the correct classification are deleted. Littlestone [19] shows that this algorithm makes at most $\lg|C_n|$ mistakes. Now suppose the prediction algorithm predicts according to the majority of concepts in set $\mathcal{V}'$, the set of all concepts in C_n consistent with all *incorrectly* predicted instances. Littlestone [19] also proves that this *space-efficient halving algorithm* makes at most $\lg|C_n|$ mistakes.

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

THEOREM 4.1. *An approximate halving algorithm makes at most* $\log_{(1-\varphi)^{-1}}|C_n|$ *mistakes for learning* C_n.

Proof. Each time a mistake is made, the number of concepts that remain in $\mathcal{V}$ are reduced by a factor of at least $1 - \varphi$. Thus after at most $\log_{(1-\varphi)^{-1}}|C_n|$ mistakes there is only one consistent concept left in C_n. □

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

When we are given an instance $x \in X_n$, one way to predict as dictated by the halving algorithm is to count the number of concepts in $\mathcal{V}$ for which $c(x) = 0$ and for which $c(x) = 1$ and then to predict with the majority. As we shall see, by using these ideas we can use an approximate counting scheme to implement the approximate halving algorithm.

We now introduce the notion of an approximate counting scheme for counting the number of elements in a finite set $\mathcal{S}$. Let x be a description of a set $\mathcal{S}_x$ in some natural encoding. An *exact counting scheme* on input x outputs $|\mathcal{S}_x|$ with probability 1. Such a scheme is polynomial if it runs in time polynomial in $|x|$. Sometimes exact counting can be accomplished in polynomial time; however, many counting problems are $\#\mathcal{P}$-complete and thus are assumed to be intractable. (For a discussion of the class $\#\mathcal{P}$ see Valiant [26].) For many $\#\mathcal{P}$-complete problems good approximations are possible [16], [24], [25]. A *randomized approximation scheme* R for a counting problem satisfies the following condition for all $\epsilon, \delta > 0$:

$$\Pr\left[\frac{|\mathcal{S}_x|}{(1+\epsilon)} \leq R(x,\epsilon,\delta) \leq |\mathcal{S}_x|(1+\epsilon)\right] \geq 1-\delta,$$

where $R(x, \epsilon, \delta)$ is R's estimate on input x, ϵ, and δ. In other words, with high probability, R estimates $|\mathcal{S}_x|$ within a factor of $1+\epsilon$. Such a scheme is *fully polynomial* if it runs in time polynomial in $|x|$, $\frac{1}{\epsilon}$, and $\lg \frac{1}{\delta}$. For further discussion see Sinclair [24].

We now review work on counting the number of linear extensions of a total order. That is, given a partial order on a set of n elements, the goal is to compute the number of total orders that are linear extensions of the given partial order. We discuss the relationship between this problem and that of computing the volume of a convex polyhedron. (For more details on this subject, see Lovász [21, §2.4].) Given a convex set S and an element a of $\Re^n$, a *weak separation oracle* either (i) asserts that $a \in S$ or (ii) asserts that $a \notin S$ and supplies a reason why. In particular, for closed convex sets in $\Re^n$, if $a \notin S$, then there exists a hyperplane separating a from S. So if $a \notin S$, the oracle responds with such a separating hyperplane as the reason why $a \notin S$.

We now discuss how to reduce the problem of counting the number of extensions of a partial order on n elements to that of computing the volume of a convex n-dimensional polyhedron given by a separation oracle. The polyhedron built in the reduction will be a subset of $[0, 1]^n$ (i.e., the unit hypercube in $\Re^n$), where each dimension corresponds to one of the n elements. Observe that any inequality $x_i > x_j$ defines a half-space in $[0, 1]^n$. Let $\Delta(t)$ denote the polyhedron obtained by taking the *intersection* of the half-spaces given by the inequalities of the partial order t. (See Fig. 8 for an example with $n = 3$.) For any pair

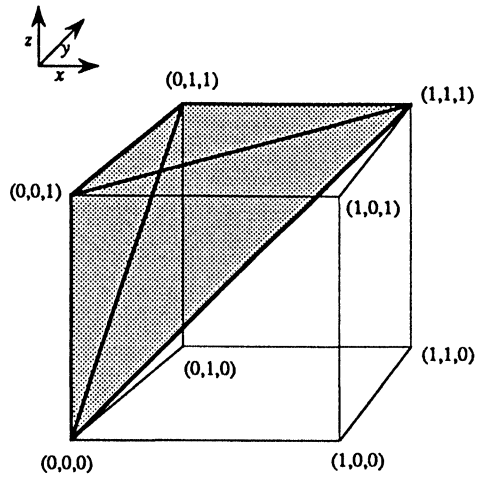


FIG. 8. *Polyhedron formed by the total order* $z > y > x$.

of total orders t_1 and t_2 the polyhedra $\Delta(t_1)$ and $\Delta(t_2)$ are simplices that intersect only in a face (zero volume): A pair of elements, say, x_i and x_j, that are ordered differently in t_1 and t_2 (such a pair must exist) define a hyperplane $x_i = x_j$ that separates $\Delta(t_1)$ and $\Delta(t_2)$. Let T_n be the set of all $n!$ total orders on n elements. Then

$$[0, 1]^n = \bigcup_{t \in T_n} \Delta(t). \tag{7}$$

In other words, the union of the polyhedra associated with all total orders yields the unit hypercube. We have already seen that polyhedra associated with the $t \in T_n$ are disjoint. To

see that they cover all of $[0, 1]^n$ observe that any point $y \in [0, 1]^n$ defines some total order t, and clearly $y \in \Delta(t)$. Let P be a partial order on a set of n elements. From equation (7) and the observation that the volumes of the polyhedra formed by each total order are equal, it follows that the volume of the polyhedron defined by any total order is $\frac{1}{n!}$. Thus it follows that for any partial order P

$$\frac{\text{number of extensions of } P}{n!} = \text{volume of } \Delta(P). \tag{8}$$

Rewriting equation (8), we obtain that

$$\text{number of extensions of } P = n! \cdot (\text{volume of } \Delta(P)). \tag{9}$$

Finally, we note that the weak separation oracle is easy to implement for any partial order. Given inputs a and S, it just checks each inequality of the partial order to see whether a is in the convex polyhedron S. If a does not satisfy some inequality, then it replies that $a \notin S$ and returns that inequality as the separating hyperplane. Otherwise, if a satisfies all inequalities, it replies that $a \in S$.

Dyer, Frieze, and Kannan [10] give a fully polynomial randomized approximation scheme (fpras) for approximating the volume of a polyhedron, given a weak separation oracle. From equation (9) we see that this fpras for estimating the volume of a polyhedron can be easily applied to estimating the number of extensions of a partial order. Furthermore, Dyer and Frieze [9] prove that it is $\#\mathcal{P}$-hard to exactly compute the volume of a polyhedron given either by a list of its facets or its vertices.

Independently, Matthews [22] has described an algorithm to generate a random linear extension of a partial order. Consider the convex polyhedron K defined by the partial order. Matthew's main result is a technique to sample nearly uniformly from K. Given such a procedure to sample uniformly from K, one can sample uniformly from the set of extensions of a partial order by choosing a random point in K and then selecting the total order corresponding to the ordering of the coordinates of the selected point. A procedure to generate a random linear extension of a partial order can then be used repeatedly to approximate the number of linear extensions of a partial order [22].

4.2. Application to learning. We begin this section by studying the problem of learning a total order under teacher-directed and self-directed learning. Then we show how to use an fpras to implement a randomized version of the approximate halving algorithm, and we apply this result to the problem of learning a total order on a set of n elements.

Under the teacher-selected query sequence we obtain an $n-1$ mistake bound. The teacher can uniquely specify the target total order by giving the $n-1$ instances that correspond to consecutive elements in the target total order. Since $n-1$ instances are needed to uniquely specify a total order, we get a matching lower bound. Winkler [28] has shown that under the learner-selected query sequence, one can also obtain an $n-1$ mistake bound. To achieve this bound the learner uses an insertion sort, as described, for instance, by Cormen, Leiserson, and Rivest [8], where for each new element the learner guesses it is smaller than each of the ordered elements (starting with the largest) until a mistake is made. When a mistake occurs, this new element is properly positioned in the chain. Thus at most $n-1$ mistakes will be made by the learner. In fact, the learner can be forced to make at least $n-1$ mistakes. The adversary gives feedback by using the following simple strategy: The first time an object is involved in a comparison, reply that the learner's prediction is wrong. In doing so, one creates a set of *chains*, where a chain is a total order on a subset of the elements. If c chains are created by this

process, then the learner has made $n - c$ mistakes. Since all these chains must be combined to get a total order, the adversary can force $c - 1$ additional mistakes by always replying that a mistake occurs the first time that elements from two different chains are compared. (It is not hard to see that the above steps can be interleaved.) Thus the adversary can force $n - 1$ mistakes.

Next we consider the case in which an adversary selects the query sequence. We first prove an $\Omega(n \lg n)$ lower bound on the number of mistakes made by any prediction algorithm. We use the following result of Kahn and Saks [17]: Given any partial order P that is not a total order there exists an incomparable pair of elements x_i, x_j such that

$$\frac{3}{11} \leq \frac{\text{number of extensions of } P \text{ with } x_i \leq x_j}{\text{number of extensions of } P} \leq \frac{8}{11}.$$

So the adversary can always pick a pair of elements, so that regardless of the learner's prediction, the adversary can report that a mistake was made while only eliminating a constant fraction of the remaining total orders.

Finally, we present a polynomial prediction algorithm making $n \lg n + (\lg e)\lg n$ mistakes with very high probability. We first show how to use an exact counting algorithm R, for counting the number of concepts in C_n consistent with a given set of examples, to implement the halving algorithm.

LEMMA 4.2. *Given a polynomial algorithm R to exactly count the number of concepts in C_n consistent with a given set E of examples, one can construct an efficient implementation of the halving algorithm for C_n.*

Proof. We show how to use R to efficiently make the predictions required by the halving algorithm. To make a prediction for an instance x in X_n the following procedure is used: Construct E^- from E by appending x as a negative example to E. Use the counting algorithm R to count the number of concepts $C^- \in \mathcal{V}$ that are consistent with E^-. Next, construct E^+ from E by appending x as a positive example to E. As before, use R to count the number of concepts $C^+ \in \mathcal{V}$ that are consistent with E^+. Finally, if $|C^-| \geq |C^+|$, then predict that x is a negative example; otherwise, predict that x is a positive example.

Clearly, a prediction is made in polynomial time, since it only requires calling R twice. It is also clear that each prediction is made according to the majority of concepts in $\mathcal{V}$. □

We modify this basic technique to use an fpras instead of the exact counting algorithm to obtain an efficient implementation of a randomized version of the approximate halving algorithm. In doing so, we obtain the following general theorem describing when the existence of an fpras leads to a good prediction algorithm. We then apply this theorem to the problem of learning a total order.

THEOREM 4.3. *Let R be an fpras for counting the number of concepts in 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 in time polynomial in n and $\lg \frac{1}{\delta}$ and makes at most $\lg |C_n|(1 + \frac{\lg e}{n})$ mistakes with probability at least $1 - \delta$.*

Proof. The prediction algorithm implements the procedure described in Lemma 4.2 with the exact counting algorithm replaced by the fpras $R(n, \frac{1}{n}, \frac{\delta}{2|X_n|})$. Consider the prediction for an instance $x \in X_n$. Let $\mathcal{V}$ be the set of concepts that are consistent with all previous instances. Let r^+ (respectively, r^-) be the number of concepts in $\mathcal{V}$ for which x is a positive (negative) instance. Let $\hat{r}^+$ (respectively, $\hat{r}^-$) be the estimate output by R for r^+ (r^-). Since R is an fpras, with probability at least $1 - \frac{\delta}{|X_n|}$

$$\frac{r^-}{1+\epsilon} \leq \hat{r}^- \leq (1+\epsilon)r^- \quad \text{and} \quad \frac{r^+}{1+\epsilon} \leq \hat{r}^+ \leq (1+\epsilon)r^+,$$

where $\epsilon = \frac{1}{n}$. Without loss of generality assume that the algorithm predicts that x is a negative instance and thus $\hat{r}^- \geq \hat{r}^+$. Combining the above inequalities and the observation that $r^- + r^+ = |V|$, we obtain that $r^- \geq \frac{|V|}{1+(1+\epsilon)^2}$.

We define an *appropriate* prediction to be a prediction that agrees with *at least* $\frac{|V|}{1+(1+\epsilon)^2}$ of the concepts in V. To analyze the mistake bound for this algorithm we suppose that each prediction is appropriate. For a single prediction to be appropriate, both calls to the fpras R must output a count that is within a factor of $1+\epsilon$ of the true count. So any given prediction is appropriate with probability at least $1 - \frac{\delta}{|X_n|}$, and thus the probability that all predictions are appropriate is at least

$$1 - |X_n|\left(\frac{\delta}{|X_n|}\right) = 1 - \delta.$$

Clearly, if all predictions are appropriate, then the above procedure is in fact an implementation of the approximate halving algorithm with $\varphi = \frac{1}{1+(1+\epsilon)^2}$, and thus by Theorem 4.1 at most $\log_{(1-\varphi)^{-1}} |C_n|$ mistakes are made. Substituting ϵ with its value of $\frac{1}{n}$ and simplifying the expression, we obtain that with probability at least $1 - \delta$

$$\text{number of mistakes} \leq \frac{\lg |C_n|}{\lg \frac{1}{1-\varphi}} = \frac{\lg |C_n|}{\lg\left(1 + \frac{n^2}{n^2+2n+1}\right)}. \tag{10}$$

Since $\frac{n^2}{n^2+2n+1} \geq 1 - \frac{2}{n}$,

$$\begin{aligned} \frac{1}{\lg\left(1 + \frac{n^2}{n^2+2n+1}\right)} &\leq \frac{1}{\lg\left(1 + 1 - \frac{2}{n}\right)} \\ &= \frac{1}{1 + \lg\left(1 - \frac{1}{n}\right)} \\ &= 1 - \frac{\lg\left(1 - \frac{1}{n}\right)}{1 + \lg\left(1 - \frac{1}{n}\right)}. \end{aligned}$$

By applying the inequalities $\lg\left(1 - \frac{1}{n}\right) \geq \frac{-\lg e}{n-1}$ and $1 + \lg\left(1 - \frac{1}{n}\right) \leq 1 - \frac{\lg e}{n}$ it follows that

$$\begin{aligned} \frac{\lg\left(1 - \frac{1}{n}\right)}{1 + \lg\left(1 - \frac{1}{n}\right)} &\geq \frac{\frac{-\lg e}{n-1}}{1 - \frac{\lg e}{n}} \\ &= \frac{-\lg e}{n - 1 - \frac{n-1}{n}\lg e} \\ &\geq \frac{-\lg e}{n}. \end{aligned}$$

Finally, applying these inequalities to equation (10) yields that

$$\text{number of mistakes} \leq \frac{\lg |C_n|}{\lg\left(1 + \frac{n^2}{n^2+2n+1}\right)} \leq \lg |C_n|\left(1 + \frac{\lg e}{n}\right). \qquad \square$$

Note that we could modify the above proof by not requiring that all predictions be appropriate. In particular, if we allow γ predictions not to be appropriate, then we get a mistake bound of $\lg |C_n|\left(1 + \frac{\lg e}{n}\right) + \gamma$.

We now apply this result to obtain the main result of this section. Namely, we describe a randomized polynomial prediction algorithm for learning a total order in the case in which the adversary selects the query sequence.

THEOREM 4.4. *There exists a prediction algorithm A for learning total orders such that on input δ (for all $\delta > 0$), and for any query sequence provided by the adversary, A runs in time polynomial in n and $\lg \frac{1}{\delta}$ and makes at most $n \lg n + (\lg e)\lg n$ mistakes with probability at least $1 - \delta$.*

Proof. We apply the results of Theorem 4.3 by using the fpras for counting the number of extensions of a partial order given independently by Dyer, Frieze, and Kannan [10] and by Matthews [22]. We know that with probability at least $1 - \delta$ the number of mistakes is at most $\lg |C_n|(1 + \frac{\lg e}{n})$. Since $|C_n| = n!$, the desired result is obtained. □

We note that the probability that A makes more than $n \lg n + (\lg e)\lg n$ mistakes does not depend on the query sequence selected by the adversary. The probability is taken over the coin flips of the randomized approximation scheme.

Thus, as in learning a k-binary-relation by using a row-filter algorithm, we see that a learner can do asymptotically better with self-directed learning than with adversary-directed learning. Furthermore, whereas the self-directed learning algorithm is deterministic, here the adversary-directed algorithm is randomized.

As a final note, observe that we have just seen how a counting algorithm can be used to implement the halving algorithm. In her thesis Goldman [11] has described conditions under which the halving algorithm can be used to implement a counting algorithm.

5. Conclusions and open problems. We have formalized and studied the problem of learning a binary relation between two sets of objects and between a set and itself under an extension of the on-line learning model. We have presented general techniques to help develop efficient versions of the halving algorithm. In particular, we have shown how a fully polynomial randomized approximation scheme 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. The specific results are summarized in Table 2. In this table all lower bounds are information-theoretic bounds and all upper bounds are for polynomial-time learning algorithms. Also, unless otherwise stated, the results listed are for deterministic learning algorithms.

From Table 2 one can see that several of the above bounds are tight and several others are asymptotically tight. However, for the problem of learning a k-binary-relation there is a gap in the bound for the random and adversary (except $k \leq 2$) directors. Note that the bounds for row-filter algorithms are asymptotically tight for k constant. Clearly, if we want asymptotically tight bounds that include a dependence on k, we cannot use only two row types in the matrix used for the projective geometry lower bound.[5]

For the problem of learning a total order all the above bounds are tight or asymptotically tight. Although the fully polynomial randomized approximation scheme for approximating the number of extensions of a partial order is a polynomial-time algorithm, the exponent on n is somewhat large and the algorithm is quite complicated. Thus an interesting problem is to find a practical prediction algorithm for the problem of learning a total order. Another interesting direction of research is to explore other ways of modeling the structure in a binary relation. Finally, we hope to find other applications of fully polynomial randomized approximation schemes to learning theory.

[5] Chen [7] has recently extended the projective geometry argument to obtain a lower bound of $\Omega(n\sqrt{m \lg k})$ for $m \geq n \lg k$.

TABLE 2
Summary of results.

Concept class	Director	Lower bound	Upper bound	Notes
Binary relation (k row types)	Learner	$\frac{km}{2} + (n - \frac{k}{2})\lfloor \lg k - 1 \rfloor$	$km + (n-k)\lfloor \lg k \rfloor$	
	Teacher	$km + (n-k)(k-1)$	$km + (n-k)(k-1)$	
	Adversary	$km + (n-k)\lfloor \lg k \rfloor$	$O(km + n\sqrt{m \lg k})$	Due to M. Warmuth[a]
	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.	$\frac{km}{2} + (n - \frac{k}{2})\lfloor \lg k - 1 \rfloor$	$O(km + nk\sqrt{H})$	Avg. case, row-filter alg.
Total order	Teacher	$n - 1$	$n - 1$	
	Learner	$n - 1$	$n - 1$	Due fo P. Winkler
	Adversary	$\Omega(n \lg n)$	$n \lg n + (\lg e) \lg n$	Randomized algorithm

[a]Note that if computation time is not a concern, we have shown that the halving algorithm makes at most $km + (n-k)\lg k$ mistakes.

Acknowledgments. The results in §4 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 3.13. We also thank Nick Littlestone, Bob Sloan, and Ken Goldman for their comments.

REFERENCES

[1] D. ANGLUIN, *Learning regular sets from queries and counterexamples*, Inform. and Comput., 75 (1987), pp. 87–106.
[2] ———, *Queries and concept learning*, Mach. Learning, 2 (1988), pp. 319–342.
[3] D. ANGLUIN AND L. G. VALIANT, *Fast probabilistic algorithms for Hamiltonian circuits and matchings*, J. Comput. System Sci., 18 (1979), pp. 155–193.
[4] J. BARZDIN AND R. FREIVALD, *On the prediction of general recursive functions*, Soviet Math. Dok., 13 (1972), pp. 1224-1228.
[5] A. BLUM, *An $\tilde{O}(n^{0.4})$-approximation algorithm for 3-coloring*, in Procceedings of the Twenth First Annual ACM Symposium on Theory of Computing, 1989, pp. 535–542.
[6] R. CARMICHAEL, *Introduction to the Theory of Groups of Finite Order*, Dover, New York, 1937.
[7] W. CHEN, personal communication, 1991.
[8] T. H. CORMEN, C. E. LEISERSON, AND R. L. RIVEST, *Introduction to Algorithms*, MIT Press/McGraw-Hill, Cambridge, MA, 1990.
[9] M. DYER AND A. FRIEZE, *On the complexity of computing the volume of a polyhedron*, SIAM J. Comput., 17 (1988), pp. 967–974.
[10] M. DYER, A. FRIEZE, AND R. KANNAN, *A random polynomial-time algorithm for approximating the volume of convex bodies*, J. Assoc. Comput. Mach., 38 (1991), pp. 1–17.
[11] S. A. GOLDMAN, *Learning Binary Relations, Total Orders, and Read-once Formulas*, Ph.D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1990.
[12] S. A. GOLDMAN AND M. K. WARMUTH, *Learning binary relations with weighted majority voting*, Proc. of the Sixth Annual Workshop on Computational Learning Theory, July 1993, to appear.
[13] T. GRADSHTEYN AND I. RYZHIK, *Tables of Integral, Series, and Products*, Academic Press, New York, 1980; corrected and enlarged edition by A. Jeffrey.

[14] D. HAUSSLER, M. KEARNS, N. LITTLESTONE, AND M. K. WARMUTH, *Equivalence of models for polynomial learnability*, Information and Computation, 95 (1991), pp. 129–161.

[15] D. HAUSSLER, N. LITTLESTONE, AND M. WARMUTH, *Expected mistake bounds for on-line learning algorithms*, unpublished manuscript, 1988.

[16] M. JERRUM AND A. SINCLAIR, *Approximating the permanent*, SIAM J. Comput., 18 (1989), pp. 1149–1178.

[17] J. KAHN AND M. SAKS, *Balancing poset extensions*, Order, 1 (1984), pp. 113–126.

[18] N. LINIAL AND U. VAZIRANI, *Graph products and chromatic numbers*, in Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Washington, DC, 1989, pp. 124–128.

[19] N. LITTLESTONE, *Learning when irrelevant attributes abound: A new linear-threshold algorithm*, Mach. Learning, 2 (1988), pp. 285–318.

[20] N. LITTLESTONE AND M. K. WARMUTH, *The weighted majority Algorithm*, in Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Washington, DC, 1989, pp. 256–261. To appear in Information and Computation.

[21] L. LOVÁSZ, *An algorithmic theory of numbers, graphs and convexity*, in CBMS–NSF Regional Conference Series on Applied Mathematics, Philadelphia, PA., 1986.

[22] P. MATTHEWS, *Generating a random linear extension of a partial order*, Ann. Prob., 19 (1991), pp. 1367–1392.

[23] R. L. RIVEST AND P. SLOAN, *Learning complicated concepts reliability and usefully*, in Proceedings of the 1988 Workshop on Computational Learning Theory, Morgan Kaufmann, San Mateo, CA, 1988, pp. 69–79.

[24] A. SINCLAIR, *Randomised Algorithms for Counting and Generating Combinatorial Structures*, Ph.D. thesis, Department of Computer Science, University of Edinburgh, Edinburgh, Scotland, U.K., 1988.

[25] L. STOCKMEYER, *An approximation algorithm for # P*, SIAM J. Comput., 14 (1985), pp. 849–861.

[26] L. VALIANT, *The complexity of computing the permanent*, Theoret. Comput. Sci., 8 (1979), pp. 198–201.

[27] ———, *A theory of the learnable*, Comm. ACM, 27 (1984), pp. 1134–1142.

[28] P. WINKLER, personal communication, 1989.