Online Learning of Binary and $n$-ary Relations over Clustered Domains

Atsuyoshi Nakamura

Internet Systems Research Laboratories, NEC Corporation, 4-1-1 Miyazaki Miyamae-ku, Kawasaki 216-8555, Japan
E-mail: atsu@ccm.cl.nec.co.jp

and

Naoki Abe

Thomas J. Watson Research Center, IBM Corporation, P. O. Box 218 Yorktown Heights, New York, 10598
E-mail: nabe@us.ibm.com

Received August 22, 2000; revised November 6, 2001

We consider the online learning problem for binary relations defined over two finite sets, each clustered into a relatively small number $k, l$ of types (such a relation is termed a $(k, l)$-binary relation), extending the models of S. Goldman, R. Rivest, and R. Schapire (1993, SIAM J. Comput. 22, 1006–1034). We investigate the learning complexity of $(k, l)$-binary relations with respect to both the self-directed and adversary-directed learning models. We also generalize this problem to the learning problem for $(k_1, \ldots, k_d)$-$d$-ary relations. In the self-directed model, we exhibit an efficient learning algorithm which makes at most $kl + (n - k)\log k + (m - l)\log l$ mistakes, where $n$ and $m$ are the number of rows and columns, roughly twice the lower bound we show for this problem, $\frac{1}{2}[\log k][\log l] + \frac{1}{2}(n - k)[\log k] + \frac{1}{2}(m - l)[\log l]$. In the adversary-directed model, we exhibit an efficient algorithm for the $(2, 2)$-binary relations, which makes at most $n + m + 2$ mistakes, only two more than the lower bound we show for this problem, $n + m$. As for $(k_1, \ldots, k_d)$-ary relations, we obtain lower bounds and upper bounds on the number of mistakes in the self-directed model, teacher-directed model, and adversary-directed model. Finally we show that, although the sample consistency problem for $(2, 2)$-binary relations is solvable in polynomial time, the same problem for $(2, 2, 2)$-ternary relations is already NP-complete.

Key Words: binary relation; online learning; mistake bound model.

---

1A preliminary version of this paper appeared in the Proceedings of the Eighth Annual Conference on Computational Learning Theory, July 1995.

2Current affiliation: Information Processing Laboratory, Research Group of Mathematical Information Engineering, Division of Systems and Information Processing, Graduate School of Engineering, Hokkaido University, Kita 13 Nishi 8, Kita-ku, Sapporo 060-8628, Japan; E-mail: atsu@main.eng.hokudai.ac.jp.

3Most of the work presented herein was performed while this author was with NEC Corporation.
1. INTRODUCTION

Binary relations are a general and basic form of knowledge, and the associated learning problem is considered to be of prime importance in many branches of artificial intelligence, such as semantic knowledge acquisition in natural language processing \[1,11\] and collaborative filtering \[9, 12\]. A binary relation $R$ between sets $A$ and $B$ can be formalized as a function of two arguments from $A \times B$ to $\{0,1\}$, defined by $R(i,j) = 1$ if and only if $R$ holds between $i$ and $j$, and can also be thought of as a $\{0,1\}$-valued matrix with $A$ being the rows and $B$ the columns.

In the computational learning theory literature, the online learning problem for binary relations was first considered by Goldman et al. \[4\] and subsequently by Goldman and Warmuth \[6\] among others, in which it was assumed that the rows can be classified into a relatively small number $k$ of types. (Such a relation was called a $k$-binary relation.) Here, two rows are said to be of the same type, if they agree in all columns. Goldman and Warmuth also addressed the problem of learning nonpure relations, in which rows of the same type are allowed to disagree in a small number of columns, but here we primarily focus on our attention on the pure relation model and extend it by classifying columns into a small number of types as well as the rows and consider the learning problem for $(k,l)$-binary relations, namely those representable by matrices having at most $k$ row types and at most $l$ column types.

Goldman et al. \[4\] considered four variants of Littlestone’s online learning model \[7\]: the randomly directed, self-directed, adversary-directed, and teacher-directed models. In this paper, we mainly consider two out of the four: the self-directed model in which the learner gets to pick the next instance to predict, and the adversary-directed model (the original model of \[7\]) in which an adversary selects the worst-case sequence of trials. In the extension to learning $d$-ary relations, we also consider the teacher-directed model in which a helpful teacher selects the next instance to predict and the prediction performance is measured using the worst case mistake bound over all consistent learners. For extensive treatment of the self-directed and teacher-directed learning models, we refer the reader to the work of Goldman and Sloan \[5\] and Goldman and Kearns \[3\], respectively.

In the self-directed model, we exhibit a learning algorithm which makes at most $kl + (n-k) \log k + (m-l) \log l$ mistakes, where $n$ is the number of rows and $m$ is the number of columns. Since the number of column types is at most $2^k$, we get a mistake bound of $km + (n-k) \log k$ for $k$-binary relations by substituting $2^k$ for $l$ in the above bound. So our bound essentially generalizes the bound shown in \[4\] for $k$-binary relations, $km + (n-k) \lfloor \log k \rfloor$.

In the adversary-directed model, we consider the learning problem for the $(2, 2)$-binary relations and exhibit a learning algorithm which makes at most $m + n + 2$ mistakes. This bound is only two more than the lower bound of $m + n$ we show for the same problem in this paper and also smaller than the mistake bound of $2m + n - 2$ for the class of 2-binary relations shown in \[4\]. As for $(k,l)$-binary relations for $k > 2$ or $l > 2$, at this point efficient algorithms with a nearly optimal mistake bound have not been found. A piece of evidence that suggests that designing such

\[^{4}\text{We briefly deal with non-pure relations in Section 4.3.2.}\]
algorithms may be challenging is the fact that the sample consistency problem\(^5\) (i.e., the problem of judging to see if there is a consistent hypothesis for an input sample) for binary relations is known to be NP-complete [4].\(^6\)

Binary relations can be naturally extended to \(d\)-ary relations between \(d\) sets,\(^7\) representable by \(n_1 \times \cdots \times n_d\) \(\{0,1\}\)-valued matrices. We consider the learning problem for the class of \((k_1,\ldots,k_d)\)-\(d\)-ary relations representable by matrices having \(k_j\) types for each dimension \(j \in \{1,\ldots,d\}\). We generalize lower bounds and upper bounds on the number of mistakes for \(k\)-binary relations shown in [4, 6].

The first lower bound we obtain is a general bound which holds for all four models considered in [4], and it is \(1/2^d \prod_{j=1}^d \log k_j + \frac{1}{2} \sum_{j=1}^d (n_j - k_j) \log k_j\), with a moderate condition on the \(k_j\). In the self-directed model, we show an upper bound of \(\prod_{j=1}^d k_j + \sum_{j=1}^d (n_j - k_j) \log k_j\) by extending our learning algorithm for \((k,l)\)-binary relations. Note that these (lower and upper) bounds are quite tight, since they are roughly within a factor of 2, when each \(k_j\) is considered to be a relatively small constant (relative to \(n_j\)).

In the teacher-directed model, we show an upper bound of \(\prod_{j=1}^d k_j + \sum_{j=1}^d (k_j - 1)(n_j - k_j)\) which coincides with the lower bound we obtained when all \(k_j\) are identical.

In the adversary-directed model, we show a lower bound of \(\prod_{j=1}^d k_j + \sum_{j=1}^d (n_j - k_j) \log k_j\), again with a moderate condition on the \(k_j\). This lower bound is also quite tight because the halving algorithm [7], which is not a polynomial time algorithm, achieves a mistake bound of \(\prod_{j=1}^d k_j + n_1 \log k_j\). Concerning mistake bounds of polynomial time algorithms, we analyzed extensions of algorithms ConsMajorityPredict [4] and Learn-Relation [6], which we call cross methods. When predicting the label of an entry, a cross method makes use of only the known entries whose coordinates differ from that entry in just one coordinate. We obtain an upper bound of \(f(\mathbf{i}, \mathbf{k})d + \sqrt{f(\mathbf{i}, \mathbf{k})} \sum_{j=1}^d ((k_j^2 - 1)/k_j^n) n_j(n_j - 1)\) for Cross-Cons Majority Predict, and an upper bound of \(f(\mathbf{i}, \mathbf{k})d + \sqrt{(\log e/e)f(\mathbf{i}, \mathbf{k})} \sum_{j=1}^d n_j(n_j - 1)\) for Cross-Learn-Relation, where \(f(\mathbf{i}, \mathbf{k}) = \prod_{j=1}^d n_j^{1/\sum_{j=1}^d (n_j/k_j)}\). We also show a lower bound for any deterministic cross method which is \(n^{d-1}\) when \(n_1 = \cdots = n_d = n\). If in addition we have \(k_1 = \cdots = k_d = k\), then \(f(\mathbf{i}, \mathbf{k})d = kn^{d-1}\) holds. This indicates that it is not possible for any deterministic cross methods to dramatically improve the prediction performance of Cross-ConsMajorityPredict and Cross-Learn-Relation.

Finally, we show a related hardness result. In particular, we show that the sample consistency problem for \((2,2,2)\)-ternary relations is NP-complete, although the same problem for \((2,2)\)-binary relations is solvable in polynomial time. These results are consistent with the fact that \((2,2)\)-binary relations are efficiently learnable in the adversary-directed model with a nearly optimal mistake bound, whereas the best known efficient online learning algorithms for \((2,2,2)\)-ternary relations (the cross methods) has worst case mistake bounds which are far from the corresponding lower bound.

\(^5\)Goldman, Rivest, and Shapire call this problem the matrix \(k\)-complexity problem.

\(^6\)We note that NP-completeness of the sample consistency problem only precludes one approach to learning the class in question, namely of finding a hypothesis consistent with the input sample.

\(^7\)General relations are usually referred to as \(n\)-ary relations, but here we use \(n\) to denote the number of rows, and let \(d\) denote the arity of the relation to avoid confusion.
ONLINE LEARNING OF BINARY AND n-ARY RELATIONS

2. PRELIMINARIES

We first define some pieces of notation we make use of in this paper. We let \( \mathbb{N} \) denote the set of natural numbers, and \( \mathbb{N}^d \) its \( d \)-fold Cartesian product. For any \( i, j \in \mathbb{N} \) such that \( i \leq j \), we define \([i, j] = \{i, i+1, \ldots, j-1, j\}\) and let \([i] = [1, i]\).

A partition \( P \) of \( X \) is a collection of mutually disjoint subsets of \( X \) such that \( \bigcup_{p \in P} p = X \). We say that partition \( P \) is coarser than partition \( Q \) (or \( Q \) is finer than \( P \)) and write \( P \preceq Q \), if \( \forall p \in P, \exists S \subseteq Q \) such that \( p = \bigcup_{s \in S} s \).

The learning model we consider in this paper is the online learning (mistake bound) model introduced by Littlestone [7]. We will explain this model in general for any concept class. Let set \( X \) be the instance domain. A concept \( c \) over \( X \) is a subset of \( X \) and a concept class \( C \) is a class of subsets of \( X \). We assume that there is a target concept \( c \in C \), which the learning algorithm is trying to learn, and each instance is labeled 1 or 0 according to whether it belongs to the target concept or not. Learning takes place in a sequence of trials. In each trial the learner is given an unlabeled instance \( x \in X \), predicts the label of \( x \), and is then told the correct label of \( x \). If the prediction is incorrect, the learner has made a mistake. The learner’s performance is measured by the total number of mistakes made for the worst case target in \( C \).

In the original online learning model it is assumed that an adversary selects the sequence of instances given to the learner, and thus the learner’s performance is evaluated in the worst case over the sequences of instances as well. Goldman et al. [4] called this model the adversary-directed model and introduced several other variants of the original model, including the self-directed model in which the learner selects the sequence of instances and the teacher-directed model in which a helpful teacher selects the sequence of instances. In the self-directed model, the learner’s performance is measured with respect to the worst case target concept in the target concept class. In the teacher-directed model, the evaluation is done in terms of the worst case number of mistakes over all concepts in the target class and all consistent learners, namely those learners whose hypotheses are always consistent with all the known instances. In this paper, we mainly consider the first two models, except in Section 4.2, in which we also consider the teacher-directed model.

A binary relation \( R \) between sets \( N_r \) and \( N_c \) is formally a function from \( N_r \times N_c \) to \([0, 1]\), defined by \( R(x, y) = 1 \) if and only if \( R \) holds between \( x \) and \( y \). In this paper, we assume without loss of generality that \( N_r = [n] \) and \( N_c = [m] \) for some \( n, m \in \mathbb{N} \), and therefore any binary relation considered here can be represented by an \( n \times m \{0, 1\}\)-valued matrix. Two rows \( i_1, i_2 \in N_r \) in a matrix \( M \) are of the same type, if \( M(i_1, j) = M(i_2, j) \) for all \( j \in N_c \). Similarly, two columns \( j_1, j_2 \in N_c \) are of the same type, if \( M(i, j_1) = M(i, j_2) \) for all \( i \in N_r \). For any \( k, l \in \mathbb{N} \), we let \( \mathcal{R}(k, l) \) denote the class of \( n \times m \{0, 1\}\)-valued matrices having at most \( k \) row types and \( l \) column types.

A \( d \)-ary relation \( R \) over \( X_1, \ldots, X_d \) is a function from \( X_1 \times \cdots \times X_d \) to \([0, 1]\). We assume that \( X_1 \times \cdots \times X_d \) equals \([n_1] \times \cdots \times [n_d] \), for some \( n_1, \ldots, n_d \in \mathbb{N} \) and a \( d \)-ary relation is represented by an \( n_1 \times \cdots \times n_d \{0, 1\}\)-valued matrix \( M \). For each dimension \( j \in [d] \), we define a \( j \)-submatrix (at \( a \)) of \( M \) to be the \((d - 1)\)-dimensional submatrix of \( M \) obtained by fixing the \( j \)th component (at \( a \)). For each dimension \( j \), the two indices \( i_j, i_j' \in [n_j] \) are of the same type, if their respective \( j \)-submatrices are identical. For \( \hat{k} = (k_1, \ldots, k_d) \in \mathbb{N}^d \), we let \( \mathcal{R}(\hat{k}) \) denote the class of
\( n_1 \times \cdots \times n_d \) \( \{0,1\} \)-valued matrices having at most \( k_j \) types of \( j \)-submatrices for each \( j \in [d] \).

The rest of the paper is organized as follows. In Section 3, we consider the learning problem for binary relations; with respect to the self-directed model in Section 3.1 and the adversary-directed model in Section 3.2. In Section 4, the extension to learning the general \( d \)-ary relations is considered, with respect to the self-directed model in Section 4.1, the teacher-directed model in Section 4.2, and the adversary-directed model in Section 4.3. Finally, we conclude with some remarks on the future work in Section 5.

3. LEARNING BINARY RELATIONS

3.1. Self-directed Learning

In this section, we present a self-directed learning algorithm for \( (k,l) \)-binary relations and analyze the worst case number of mistakes it makes. Goldman et al. [4] studied the self-directed learning problem for \( k \)-binary relations. Let us say that the representative row of a row type is the first row that is encountered among all rows of that row type. To predict the value of entry \((i,j)\), their algorithm employs majority vote over the column \( j \) entries of all consistent representative rows that have been encountered so far. In the worst case, their algorithm may make a mistake for every column when the row is of unknown type. We may be able to do better if we make use of the similarities between the columns. For example, even if row \( i \) is of an unknown type, we could predict the value of entry \((i,j)\) correctly if we knew that column \( j \) and column \( j' \) are of the same type and knew the value of entry \((i,j')\). Based on this intuition, we propose algorithm SD-predict that takes advantage of both column similarities and row similarities.

It may appear as if a more straightforward extension of the algorithm used in Goldman et al.’s proof would work here, but this is not the case. A crucial point in their proof is that, when predicting elements of a nonrepresentative row, every preceding representative row has been identified as a representative row. But, in order to ensure this, the prediction must be done in a raw-major order, that is, all entries of a row must be selected before proceeding to the next row. Then, to straightforwardly generalize their proof idea to both rows and columns, the sequence must be selected both in row major and column major order, which is impossible.

Suppose that \( M \in \mathcal{M}_{(k,l)} \) represents the target binary relation, defined over \( N_r \times N_c = [n] \times [m] \). Let \( P_s^M \) and \( P_c^M \) denote the partitions on the rows \( N_r \) and the columns \( N_c \) of \( M \), induced by the row and column types, respectively. Our algorithm, which we call SD-predict and is shown in Fig. 1, keeps at any point an observation matrix \( M_w \) and work partitions \( P_s^w \) and \( P_c^w \). Matrix \( M_w \) simply stores all the entries in \( M \) it has already seen, and it contains an * for all unknown entries of \( M \). Partition \( P_c^w \times P_s^w \) is meant to be the algorithm’s current hypothesis for \( P_c^M \times P_s^M \), and it is initially set to \( \{ N_r \} \times \{ N_c \} \) and becomes finer as learning proceeds. For notational convenience, for any subset \( \mathcal{J} \subseteq N_r \times N_c \), we let \( M_w(\mathcal{J}) \) denote the set \( \{ M_w(i,j) : (i,j) \in \mathcal{J} \} \).
Algorithm SD-predict

Definitions:

- $M_w$: Observation matrix (updated after each trial)
- $P_r^w \times P_c^w$: Work partitions
- $I_b(j)$: Unique member of $P_b^w$ that contains $j$.
- $S_c(i, j) = \text{set of entries of row } i \text{ contained in } I_c(j)$
- $S_r(i, j) = \text{set of entries of column } j \text{ contained in } I_r(i)$

Initialization: $P_r^w \times P_c^w = \{N_r\} \times \{N_c\}$.

Repeat the following 3 phases until there are no more unknown entries.

1. Pick an unknown entry $(i_0, j_0)$ in the current observation matrix $M_w$. Predict all unknown entries $(i, j)$ in $S_c(i_0, j_0)$ one by one as follows:

   If there are no known entries in $S_c(i_0, j_0)$,
   - then if $M_w(I_r(i_0) \times I_c(j_0))$ contains a (unique) non-\*$ value $x$
     - then predict with $x$,
     - else predict at random,
   - else predict by majority vote over the known entries in $S_c(i_0, j_0)$
     (predict at random in the case of a draw).

   If mistakes were made, let $(i_0, j_1)$ denote the entry at which the first mistake was made.

2. If $M_w(S_c(i_0, j_0))$ contains both 0 and 1, then divide $I_c(j_0)$ into $\{j \in I_c(j_0) : M_w(i_0, j) = 0\}$ and $\{j \in I_c(j_0) : M_w(i_0, j) = 1\}$, and update the work partition $P_c^w$ accordingly.

3. If $M_w(S_r(i_0, j_1))$ contains both 0 and 1, then predict all unknown entries in $S_r(i_0, j_1)$ by majority vote and divide $I_r(i_0)$ if necessary, as in Phase 2.

**FIG. 1.** Self-directed learning algorithm for binary relations.
At any point during a learning session of SD-predict, we can assume inductively that $P_r^b \geq P_b^M$ for $b \in \{r,c\}$, and that partition $P_r^w \times P_c^w$ of $N_r \times N_c$ is consistent with $M_w$, namely $M_w(\mathcal{F})$ does not contain both 0 and 1 for any $\mathcal{F} \in P_r^w \times P_c^w$. We define work partition $I_r(j)$ to be the unique member of $P_r^c$ that contains column $j$ and define $S_r(i,j)$ to be the set of entries of row $i$ contained in $I_r(j)$. Similarly, we also define work partition $I_c(i)$ to be the unique member of $P_r^r$ that contains row $i$ and define $S_c(i,j)$ the set of entries of column $j$ contained in $I_c(i)$.

Each iteration of SD-predict works in three phases. Assume that it is at the beginning of an iteration now and that $P_r^w \times P_c^w$ is consistent with $M_w$. In its first phase, it picks an unknown entry $(i_0,j_0)$ and predicts all unknown entries $(i_0,j)$ in $S_r(i_0,j_0)$ one by one as follows. If there are no known entries in $S_r(i_0,j)$, SD-predict predicts with the unique non-* value in $M_w(I_r(i_0) \times I_c(j))$ if one exists and predicts at random otherwise. If $S_r(i_0,j)$ does contain known entries, then SD-predict predicts by majority vote over them (and predicts at random in the case of a draw). If SD-predict made some mistakes in this phase, then let $(i_0,j_1)$ denote the first entry at which SD-predict made a mistake.

Having predicted all entries in $S_r(i_0,j_0)$, SD-predict goes into the second phase. If at this point $M_w(S_r(i_0,j_0))$ contains both 0 and 1, then it further divides $I_r(j_0)$ according to their values, i.e., into $\{ j \in I_r(j_0) : M_w(i_0,j) = 1 \}$ and $\{ j \in I_r(j_0) : M_w(i_0,j) = 0 \}$. Thus the partition $P_r^w$ becomes finer. (See Fig. 2.) If $S_r(i_0,j_1)$ contains both 0 and 1, then SD-predict predicts all unknown entries in $S_r(i_0,j_1)$ (this is the third phase) and divides $I_r(i_0)$ if necessary, analogously to the way it did $I_r(j_0)$. After this, $P_r^w \times P_c^w$ is guaranteed to be consistent with $M_w$, again, and $P_r^w \geq P_b^M$ holds for $b \in \{r,c\}$.

SD-predict repeats the above process with the updated values of $M_w$ and $P_r^w \times P_c^w$, until there are no more unknown entries left.

**Theorem 3.1.** For an arbitrary target matrix $M \in \mathcal{M}(k,l)$, self-directed learning algorithm SD-predict makes at most $kl + (n-k) \log k + (m-l) \log l$ mistakes.

**Proof.** In bounding the worst case number of mistakes made by SD-predict, we introduce the notion of $b$-cost for mistakes made by SD-predict in predicting the entries in each $S_b(i,j)$ for both $b = r,c$, which is meant to capture the number of

![Diagram](image.png)

**FIG. 2.** An example of how observation matrix $M_w$ and work partitions $P_r^w \times P_c^w$ are updated during a run of SD-predict.
Lemma 3.1 bounds the total \( r \)-cost and \( c \)-cost incurred during a learning session.

Precise definition of \( c \)-cost for \( S_b(i,j) \) is the number of mistakes made by SD-predict in the process for \( S_b(i,j) \) minus the number of true partitions which has nonempty intersection with \( S_b(i,j) \) and have no known entry at the beginning of the process for \( S_b(i,j) \). The definition of \( r \)-cost for \( S_b(i,j) \) is similar. For example, \( c \)-cost for \( S_b(2,1) \) in Fig. 2 is at most 0 because the number of mistakes is at most two and the number of true partitions whose entries have been selected for the first time is two. Similarly, \( r \)-cost for \( S_b(2,2) \) is 1 (= 1 – 0).

We claim that if the \( c \)-cost is positive for \( S_b(i,j) \), then \( I_r(j) \) will be divided by SD-predict into two partitions after the process for \( S_b(i,j) \) and that if the \( r \)-cost is positive for \( S_b(i,j) \), then \( I_c(i) \) will be divided. Thus if the \( b \)-cost is positive for \( S_b(i,j) \), then we can show that \( S_b(i,j) \) contains both 0-entries and 1-entries. Suppose that all entries in \( S_b(i,j) \) have identical values. Then SD-predict makes no mistakes for \( S_b(i,j) \) except at its first prediction, and so the \( b \)-cost is at most zero if there are true partitions whose entries have been selected for the first time. If all entries in \( S_b(i,j) \) have identical values, then SD-predict must be in its first phase, and thus \( P^r_b \times P^c_b \) must be consistent with \( M_u \) prior to making the predictions for \( S_b(i,j) \). So if each entry in \( S_b(i,j) \) belongs to one of the partitions whose entries have already appeared, SD-predict does not make a mistake even for the first prediction. Therefore, the \( b \)-cost for \( S_b(i,j) \) cannot be positive when \( S_b(i,j) \) contains only 0-entries or only 1-entries.

By the above argument, we see that the total \( b \)-cost can be bounded above by the total sum of \( b \)-costs for all \( S_b(i,j) \) that are divided by SD-predict. In order to bound the total \( b \)-cost above, we use two lemmas, Lemma 3.1 and Lemma 3.2. Lemma 3.1 bounds the \( b \)-cost for each \( S_b(i,j) \) and Lemma 3.2 bounds the total sum of them. Lemma 3.1 claims that the \( b \)-cost of any \( S_b(i,j) \) is at most \( \min \{u, 2u_0 + 1, 2u_1 + 1\} - h \), where \( u \) is the number of entries in \( S_b(i,j) \) and \( u_0 \) and \( u_1 \) are the number of 0-entries and 1-entries, respectively, in \( S_b(i,j) \) with respect to true matrix \( M \) and \( h \) is the number of true partitions which has nonempty intersection with \( S_b(i,j) \). The set of the pairs of the number of elements and the number containing true partitions for all work partitions in \( P^r_b \times P^c_b \) are initially \( \{(|N_b|, K_b)\} \) and finally \( \{(|I|, 1) : I \in P^M_b \} \). (Here we let \( K_r = k \) and \( K_c = l \).) Whenever \( S_b(i,j) \) is divided, the corresponding number pair \((u, h)\) is divided into \((u_0, h_0)\) and \((u_1, h_1)\) with the \( b \)-cost of at most \( \min \{u, 2u_0 + 1, 2u_1 + 1\} - h \). Thus, the total \( b \)-cost is bounded above by the sum of these \( b \)-costs throughout the process starting from \( \{(|N_b|, K_b)\} \), and ending to \( \{(|I|, 1) : I \in P^M_b \} \). Lemma 3.2 bounds this total \( b \)-cost by \( (|N_b| - K_b) \log K_b \).

Since the total number of mistakes made by SD-predict is at most the summation of all \( r \)-costs and \( c \)-costs plus the number of elements in \( P^r_b \times P^c_b \), this bound of the total \( b \)-cost implies the statement of the theorem.

**Lemma 3.1.** The \( b \)-cost for any \( S_b(i,j) \) is at most \( \min \{u, 2u_0 + 1, 2u_1 + 1\} - h \), provided that SD-predict has made mistakes predicting the entries in \( S_b(i,j) \), where \( u_0 \)
and $u_1$ are the number of 0-entries and 1-entries, respectively, in $S_b(i,j)$ with respect to true matrix $M$, and $h$ is the number of true partitions which has nonempty intersection with $S_b(i,j)$.

Proof. Let $m$ denote the number of mistakes SD-predict made for $S_b(i,j)$ and $h'$ denote the number of true partitions whose entries have been selected for the first time in the process for $S_b(i,j)$. Since $b$-cost is $m - h'$, we only have to show that $m \leq \min \{u, 2u_0 + 1, 2u_1 + 1\} - (h - h')$. Let $v$ be the number of known entries in $S_b(i,j)$ before predictions for $S_b(i,j)$ are made. Since $v \geq h - h'$, we show that $m \leq \min \{u, 2u_0 + 1, 2u_1 + 1\} - v$. It is clear that $m \leq u - v$. Here we only show that $m \leq 2u_0 + 1 - v$, because $m \leq 2u_1 + 1 - v$ can be shown similarly. When $u_0 \geq u_1$, then $2u_0 + 1 > u$ holds, and thus $m \leq u - v < 2u_0 + 1 - v$. So assume that $u_0 < u_1$. Let $v_0$ and $v_1$ be the number of known entries in $S_b(i,j)$ with value 0 and 1, respectively. Since SD-predict predicts by majority vote among the known entries in $S_b(i,j)$, it makes mistakes for at most $u_0 - v_0$ 0-valued entries and at most $u_0 + 1 - v_1$ 1-valued entries. Thus, $m \leq 2u_0 + 1 - (v_0 + v_1) = 2u_0 + 1 - v$ holds. □

Lemma 3.2. Let $G$ be $\{(u, h)\}$ initially, where $u$ and $h$ are arbitrary natural numbers satisfying $u \geq h$. Then repeat the following operation on $G$ until it is no longer possible: Replace some $(v, g)$ in $G$ by $(v_0, g_0)$ and $(v_1, g_1)$ such that $v_0, v_1, g_0, g_1$ are natural numbers satisfying $v_0 + v_1 = v$, $g_0 + g_1 = g$, $v_0 \geq g_0$, and $v_1 \geq g_1$. Define the cost of the replacement to be $\min \{v, 2v_0 + 1, 2v_1 + 1\} - g$.

Then the total cost is at most $(u - h) \log h$.

Proof. We prove this by induction on $h$. When $h = 1$, no operation can be applied on $G$, so the total cost is 0, and thus the statement of the lemma holds in this case. So suppose that $h \geq 2$. Assume the statement of the lemma holds for all $G = \{(u', h')\}$ with $h' < h$. Assume that $(u, h)$ is replaced by $(u_0, h_0)$ and $(u_1, h_1)$. By the inductive hypotheses on $(u_0, h_0)$ and $(u_1, h_1)$, the total cost for $G = \{(u, h)\}$ is at most $f(u_0, h_0) \equiv (u_0 - h_0) \log h_0 + (u_1 - h_1) \log h_1 + \min \{u, 2u_0 + 1, 2u_1 + 1\} - h$. We will show below that $f(u_0, h_0) \leq (u - h) \log h$ holds for all choices of $(u_0, h_0)$.

1. When $u_0 = u_1$.

Since $\log x$ is concave and $x \log x$ is convex,

$$f(u_0, h_0) = \frac{u}{2} (\log h_0 + \log h_1) - (h_0 \log h_0 + h_1 \log h_1) + u - h$$

$$\leq u \log \frac{h}{2} - h \log \frac{h}{2} + u - h = (u - h) \log h.$$  

2. When $u_0 < u_1$.

Note that in this case we have $f(u_0, h_0) = (\log h_0 - \log h_1 + 2)u_0 - h_0 \log h_0 + (u - h_1) \log h_1 + 1 - h$, using the fact that $\min \{u, 2u_0 + 1, 2u_1 + 1\} = 2u_0 + 1$.

(i) When $\log h_0 - \log h_1 + 2 \leq 0$. 

In this case, \( f(u_0, h_0) \) is decreasing in \( u_0 \) and since \( u_0 \geq h_0 \), we have
\[
f(u_0, h_0) \leq f(h_0, h_0) = (u - h) \log h_1 + 2h_0 + 1 - h \\
\leq (u - h) \log h.
\]

(ii) When \( \log h_0 - \log h_1 + 2 > 0 \).

In this case, \( f \) is increasing in \( u_0 \) and thus \( f(u_0, h_0) \leq f(\frac{u_0 - 1}{2}, h_0) \) holds. Now if we define \( g(h_0) = f(\frac{u_0 - 1}{2}, h_0) \) then \( g(h_0) \) is increasing for \( h_0 \leq \frac{h_1}{2} \) and decreasing for \( h_0 \geq \frac{h}{2} \), and \( g(\frac{h_1}{2}) \leq g(\frac{h}{2}) \) holds because
\[
g'(h_0) = \log \frac{h - h_0}{h_0} + \frac{u(h - 1 - 2h_0) + u - h}{2h_0(h - h_0)}.
\]

Since there is no integer between \( \frac{h_1}{2} \) and \( \frac{h}{2} \), this means that the maximum value of \( g \) assumed by an integer is at most \( g(\frac{h}{2}) \). Hence we have
\[
f(u_0, h_0) \leq f \left( \frac{u_0 - 1}{2}, \frac{h}{2} \right) = (u - h) \log \frac{h}{2} + u - h \\
= (u - h) \log h.
\]

3. When \( u_0 > u_1 \).
Since function \( f \) is symmetric with respect to \((u_0, h_0)\) and \((u_1, h_1)\), the argument for the case \( u_0 < u_1 \) applies.

This completes the proof of Theorem 3.1. \( \blacksquare \)

Since any matrix in \( \mathcal{M}_{(k,l)} \) has at most \( 2^l \) row types and \( 2^k \) column types, we can assume the following condition without loss of generality:
\[
\log \max\{k, l\} \leq \min\{k, l\}. \tag{1}
\]

Let \( \mathcal{M}_{(k,*)} \) denote the class of all matrices having at most \( k \) row types; i.e., \( \mathcal{M}_{(k,*)} = \mathcal{M}_{(k,2^l)} \). Since function \( f(l) = kl + (n - k) \log k + (m - l) \log l \) increases for \( l \leq \min\{2^k, m\} \), the mistake bound of \( km + (n - k) \log k \) obtained from the upper bound of Theorem 3.1 by substituting \( 2^k \) or \( m \) for \( l \) is an upper bound for \( \mathcal{M}_{(k,*)} \).

Note that this bound is almost the same as the bound \( km + (n - k) \lfloor \log k \rfloor \) shown in Theorem 2 in [4].

Note also that from the general lower bound for all four models (self-directed, adversary-directed, teacher-directed, and randomly directed models) given in Corollary 4.1 which we prove later and (1), we can obtain a lower bound of \( \frac{1}{4} \lfloor \log k \rfloor \lfloor \log l \rfloor + \frac{1}{2} (n - k) \lfloor \log k \rfloor + \frac{2}{3} (m - l) \lfloor \log l \rfloor \) when \( k, l \geq 10 \). Thus, the upper bound shown in Theorem 3.1 is roughly within a factor of 2 of the general lower bound when \( k \) and \( l \) are relatively small constants compared to \( n \) and \( m \).

3.2. Adversary-Directed Learning

Recall that Goldman et al. [4] have shown that the sample consistency problem for \( \mathcal{M}_{(3,*)} \) is \( \mathcal{NP} \)-complete, and hence designing efficient algorithms with a nearly
Algorithm $\mathcal{M}(2,2)$-predict

Perform the following three steps for each trial:

1. Receive an entry $(i, j)$ to be predicted

2. If X-predict has not made a mistake in the past trials:
   
   If X-predict$(i, j) = *$ then
   
   Predict with L-predict$(i, j)$
   
   else
   
   Predict with X-predict$(i, j)$
   
   Otherwise:
   
   Predict with L-predict$(i, j)$

3. Receive correct label $M(i, j)$ and let $M_w(i, j) = M(i, j)$

FIG. 3. Adversary-directed learning algorithm for $\mathcal{M}(2,2)$. 

optimal mistake bound for the class in the adversary-directed model appears to be a challenging task. Here, we exhibit an efficient learning algorithm for $\mathcal{M}(2,2)$ with a nearly optimal mistake bound. We say that a matrix in $\mathcal{M}(2,2)$ is of L-type when one type of rows (and of columns) contains both 0 and 1, and the other type of rows (and of columns) contains only 0 or only 1. We say that a matrix in $\mathcal{M}(2,2)$ is of X-type if there are two row types and two column types and both types of rows (and of columns) contain both 0 and 1. If a matrix in $\mathcal{M}(2,2)$ is neither of X-type nor of L-type, then either the rows or the columns must have only one type. Note that if a matrix is not of L-type, then two rows (columns) of different types are complements of each other. Thus, two rows (columns) are of the same type when there exists a column (row) at which they have the same value if a matrix is not of L-type.

Our algorithm for $\mathcal{M}(2,2)$, which we call $\mathcal{M}(2,2)$-predict and is shown in Fig. 3, makes use of two subsidiary learning algorithms, X-predict, which works well when the target matrix is not of L-type, and L-predict, which works well when the target is not of X-type. At the beginning of a learning session, $\mathcal{M}(2,2)$-predict predicts using X-predict provided it returns a prediction and predicts using L-predict when X-predict returns * which stands for don’t know, until it makes a mistake with a non-* value returned by X-predict. It then switches to L-predict. We can bound the worst case number of mistakes made by $\mathcal{M}(2,2)$-predict as follows.
Theorem 3.2. For all target matrices $M \in \mathcal{M}_{2,2}$, $\mathcal{M}_{2,2}$-predict makes at most $m + n + 2$ mistakes.

Before we prove Theorem 3.2, we will describe the two subsidiary algorithms, X-predict and L-predict. Let $M_w$ denote the observation matrix defined as before. For $M(i, j)$, X-predict, which is shown in Fig. 4, predicts using the values of $M_w$ of row $i$ and those of column $j$. If there exists row $i'$ such that rows $i$ and $i'$ are known to be of the same type (or different types) and $M_w(i', j) \neq *$, then X-predict predicts with $M_w(i', j)$ (or its complement). Also if there exists a column $j'$ satisfying the corresponding condition for columns, then X-predict predicts similarly using $M(i, j')$. Otherwise, X-predict outputs $* *$.

Algorithm X-predict$(i, j)$

If $\exists i'$ s.t. $M_w(i', j) \neq *$

If row $i'$ and row $i$ are known to be of the same type

(i.e., there exists a column at which they have the same value)

Output $M_w(i', j)$

If row $i'$ and row $i$ are known to be of different types

(i.e., there exists a column at which they have different values)

Output $\overline{M_w(i', j)}$

If $\exists j'$ s.t. $M_w(i, j') \neq *$

If column $j'$ and column $j$ are known to be of the same type

(i.e., there exists a row at which they have the same value)

Output $M_w(i, j')$

If column $j'$ and column $j$ are known to be of different types

(i.e., there exists a row at which they have different values)

Output $\overline{M_w(i, j')}$

Otherwise output $*$

FIG. 4. An algorithm which works well when the target is not of L-type.

*Note that this algorithm is a reliable learner [10] in that it returns $* *$ when it is not sure, and whenever it actually outputs a prediction it is never wrong.
In order to efficiently find out whether two rows (or two columns) are known to be of the same type, known to be of different types, or their relationship is unknown, X-predict can make use of graph $G_r$ (or $G_c$) described below. The vertices $V_r$ of $G_r$ are the rows, i.e., $V_r = N_r$, and $G_r$ has two kinds of edges. First, for each set of rows that are known to be of the same type, one representative $i_0$ is picked and a directed edge is placed from every vertex in that set of $i_0$. Thus, two rows are known to be of the same type if and only if their representative rows coincide. For each pair $(i_1,i_2)$ of representative rows that are known to belong to different types, an undirected edge is placed between $i_1$ and $i_2$. Thus two rows are known to be of different types if and only if there is an undirected edge between their representatives. Graph $G_c$ for columns is defined analogously. (See Fig. 5 for an example of $G_r$.)

**Lemma 3.3.** For all target matrices $M \in \mathcal{M}_{(2,2)}$ that are not of $L$-type, X-predict returns ‘*’ at most $m + n - 1$ times and never makes mistakes when it returns 0 or 1.

**Proof.** Recalling that in any matrix that is not of type L, two rows belonging to different types are complements of each other and two rows of the same type agree in every column (and the analogous statements hold for the columns), it is easy to see that X-predict never makes a mistake when it returns a non-‘*’ value.

Let $M_w(i,N_c)$ denote the set of non-‘*’ entry values in row $i$ of $M_w$, and let $M_w(N_r,j)$ denote the same for column $j$ in $M_w$. We let $h_0$ denote the number of mistakes made by X-predict when both $M_w(i,N_c)$ and $M_w(N_r,j)$ are $\emptyset$, and $h_1$ the number of mistakes made when only one of them is $\emptyset$, and $h_2$ the number of mistakes made when neither of them is $\emptyset$. Note that

$$2h_0 + h_1 \leq m + n.$$  \hspace{1cm} (2)

When a prediction mistake is made on $M(i,j)$ with $M_w(i,N_c) = \emptyset$, $M_w(N_r,j) \neq \emptyset$, $i$ becomes connected with all $i'$ such that $M_w(i',j) \neq *$ in $G_r$, so the number of connected components in $G_r$ decreases by at least one. When a mistake is made with $M_w(i,N_c) \neq \emptyset$ and $M_w(N_r,j) = \emptyset$, the number of connected components decreases by at least one in both $G_r$ and $G_c$.

There are $m + n$ connected components initially (all together in $G_r$ and $G_c$) and two connected components in the end, so we must have

$$h_1 + 2h_2 \leq m + n - 2.$$  \hspace{1cm} (3)

Combining (2) and (3), we obtain $h_0 + h_1 + h_2 \leq m + n - 1$.  \hspace{1cm} \blacksquare
Algorithm L-predict(i, j)

Let $M_w(i, N_r)$ be the set of non-'*' values in row $i$ of $M_w$.

Let $M_w(N_r, j)$ be the set of non-'*' values in column $j$ of $M_w$.

$S_w = \{M_w(i, j) \in \{0, 1\} : M_w(i, N_r) = M_w(N_r, j) = \{0, 1\}\}$

$L^w_r(i) = M_w(i, N_r) \cup S_w$

$L^w_c(j) = M_w(N_r, j) \cup S_w$

For $x \in \{0, 1\}$, let $N_x$ be the number of sets in $\{L^w_r(i), L^w_c(j)\}$ containing $x$.

If $N_0 > N_1$ then output $0$

If $N_0 < N_1$ then output $1$

Otherwise

If $S_w = \emptyset$ then

Output 0 or 1 at random

Else

Output the complement of the unique element in $S_w$

FIG. 6. An algorithm which works well when the target is not of X-type.

Next, we will describe L-predict, which is shown in Fig. 6. For a $\{*, 0, 1\}$-valued matrix $M$, let $M(i, N_r)$ denote the set of non-"*" values in row $i$ of $M$, and let $M(N_r, j)$ denote the set of non-"*" values in column $j$ of $M$. Every $\{0, 1\}$-valued L-type matrix $M \in \mathcal{M}(2,2)$ can be further classified into 0-L-type or 1-L-type, depending on whether there exists a row $i$ satisfying $M(i, N_r) = \{0\}$ or $\{1\}$. Note that L-predict can conclude that the target matrix $M$ is $x$-L-type from the values of the entries already seen, namely from the observation matrix $M_w$, provided that there is an entry $(i, j)$ such that $M_w(i, j) = \bar{x}$ and $M_w(i, N_r) = M_w(N_r, j) = \{0, 1\}$. Define $S_w$ as $S_w = \{M_w(i, j) \in \{0, 1\} : M_w(i, N_r) = M_w(N_r, j) = \{0, 1\}\}$. Note that, when it is found that the target matrix $M$ is $x$-L-type, set $S_w$ changes from $\emptyset$ to $\{x\}$. Let $L^w_r(i) = M_w(i, N_r) \cup S_w$ and $L^w_c(j) = M_w(N_r, j) \cup S_w$. (See Fig. 7.) Prior to finding out that the target matrix $M$ is $x$-L-type ($M$ can be a member of $\mathcal{M}(2,1)$ or $\mathcal{M}(1,2)$ as well) for $x = 0$ or $1$, L-predict predicts $M(i, j)$ by majority vote over the multiset obtained by taking the multiset-union of the sets $L^w_r(i)$ and $L^w_c(j)$ and predicts at random in the case of a draw. Once L-predict has found out that $M$ is $x$-L-type, it
also predicts $M(i, j)$ by majority vote over multiset obtained by taking the multiset-union of the sets $L_w^+(i)$ and $L_w^-(j)$, but predicts with $\bar{x}$ in the case of a draw. Note that L-predict never makes a mistake when $L_w^+(i) = M(i, N_r)$, $L_w^-(j) = M(N_r, j)$ and $S_w \neq \emptyset$.

**Lemma 3.4.** For all target matrices $M \in \mathcal{M}_{2,3}$ that are not of $X$-type, L-predict makes at most $m + n + 1$ mistakes.

**Proof.** We only consider the case when $M$ is of L-type. In the other cases, this theorem is proved similarly. Let $M$ be of x-L-type. Let $t_0$ be the trial number such that $S_w$ becomes $\{x\}$ at the $t_0$th prediction. Let $h_0, h_1$ be the number of mistakes made before and after the $t_0$th trial, respectively (i.e., $t < t_0$ and $t > t_0$). Let $T = |\{i : L_w^+(i) = M(i, N_r)\}| + |\{j : L_w^-(j) = M(N_r, j)\}|$. We consider the amount by which $T$ increases when a mistake is made. Note that no mistake is made after $T$ reaches $m + n$. We claim that, for a certain integer $x_0$, $T$ increases by at least $h_0 - x_0$ before the $t_0$th trial, by at least $x_0$ at the $t_0$th prediction, and by at least $h_1$ after the $t_0$th trial. Therefore, the total increase of $T$ is at least $h_0 + h_1$, which is bounded from above by $m + n$. Thus, the total number of mistakes is at most $m + n + 1$ even if we include the mistake made at the $t_0$th trial.

After the $t_0$th trial, mistakes are made only when $(L_w^+(i) = \{x\}$ or $L_w^-(j) = \{x\}$) and $M(i, j) = \bar{x}$. In this case, $L_w^+(i)$ or $L_w^-(j)$ becomes $\{x, \bar{x}\}$. Thus, every mistake forces $T$ to increase by at least 1, and hence $T$ increases by at least $h_1$ during the trials after the $t_0$th trial.

Consider the case $t \leq t_0$. We consider the number $x_0$ of times a mistake is made and $T$ does not increase, for $t < t_0$. Such a case happens only when $M(i, N_r) = M(N_r, j) = \{0, 1\}$ and $(L_w^+(i) = \emptyset$ or $L_w^-(j) = \emptyset)$. In this case, $L_w^+(i)$ or $L_w^-(j)$ remains $\{\bar{x}\}$ while $t < t_0$, because, if both of them become $\{0, 1\}$, then $S_w = \{\bar{x}\}$, which contradicts the assumption that $t < t_0$. Thus, at least $x_0$ of $L_w^+(i)$ or $L_w^-(j)$ become $\{0, 1\}$ at the $t_0$th trial. This means that $T$ increases by at least $x_0$ at the $t_0$th trial. Therefore, $T$ increases by at least $h_0 - x_0$ when $t < t_0$, and by at least $x_0$ at the $t_0$th prediction. $\blacksquare$
Proof of Theorem 3.2. If $M$ is not of L-type, then $\mathcal{M}_{(2,2)}$ predict makes at most $m + n - 1$ mistakes by Lemma 3.3. If $M$ is of L-type, then it can happen only once that $\mathcal{M}_{(2,2)}$ predict makes a mistake using a prediction that is different from that of L-predict. Thus, $\mathcal{M}_{(2,2)}$ predict makes at most $(m + n + 1)$ mistakes by Lemma 3.4.

By Theorem 4.5 we can get a lower bound of $m + n$. So, the upper bound shown in Theorem 3.2 is only 2 more than the optimal.

4. LEARNING GENERAL D-ARY RELATIONS

In this section we consider the online learning problem for d-ary relations in general. For notational convenience, we allow $a_{d+1}$ and $a_{d+2}$ to denote $a_1$ and $a_2$, respectively, for any d-dimensional vector $(a_1, \ldots, a_d)$. For example, in Theorem 4.1, the condition $l_{j+1} + 2^i \leq k_{j+1}$ for $j = d$ means $l_1 + 2^i \leq k_1$.

4.1. Self-directed Learning

4.1.1. A General Lower Bound. The next theorem is the generalization of Theorem 1 in the paper by Goldman et al. [4]. In order to prove the theorem, they considered the adversary who returns the opposite value to the learner’s prediction for all entries in $(1 - \beta)k$ rows and $\lceil \log \beta k \rceil$ columns and returns 0’s for all other entries. The point here is that any matrix having 0-entries except for $n_j$ rows and $\lceil \log \beta k \rceil$ columns has at most $k$ row types because $(1 - \beta)k + 2\log \beta k \leq k$. We generalize their idea in obtaining the next theorem.

**Theorem 4.1.** Let $\vec{l} = (l_1, \ldots, l_d) \in \mathbb{N}^d$ be a vector satisfying $l_{j+1} + 2^i \leq k_{j+1}$ for all $j \in [d]$. Any prediction algorithm for $M \in \mathcal{A}_k$ makes at least $F(\vec{l}) = \prod_{j=1}^d l_j + \sum_{j=1}^d l_j(n_j - 1 - l_{j+1})$ mistakes in the worst case, regardless of the query sequence.

**Proof.** The proof of this theorem is similar to that of Theorem 1 in [4]. Let $D = \{l_1\} \times \cdots \times \{l_d\}$ and $D_j = \{(i_1, \ldots, i_d) : i_j \in [l_j], \ l_{j+1} \in [l_{j+1} + 1, n_{j+1}], \ (\forall h \neq j, j + 1) \vec{i}_h = 1\}$. (see Fig. 8.) We claim that any matrix $M$ with 0’s for all entries except those in $D \cup \bigcup_{j=1}^d D_j$ belongs to $\mathcal{A}_k$. For each $j \in [d]$, every entry $(i_1, \ldots, i_d)$ in $D \cup \bigcup_{j=1}^d D_j$ with $i_{j+1} > l_{j+1}$ is contained only in $D_j$. So the set of $(j + 1)$-submatrices at $a \in [l_{j+1} + 1, n_{j+1}]$ is partitioned into at most $2^i$ types, and thus it follows from the condition $l_{j+1} + 2^i \leq k_{j+1}$ that the set of all $(j + 1)$-submatrices is classified into at most $k_{j+1}$ types. For each entry in $D \cup \bigcup_{j=1}^d D_j$, the adversary returns the opposite value to the learner’s prediction and returns 0’s for all other entries. It is easy to see that the number of entries in $D \cup \bigcup_{j=1}^d D_j$ is $F(\vec{l})$.

Note that the lower bound $(1 - \beta)km + n\lceil \log \beta k \rceil - (1 - \beta)k\lceil \log \beta k \rceil$ for k-binary relations in Theorem 1 [4] can be obtained from Theorem 4.1 by considering $((1 - \beta)k, \lceil \log \beta k \rceil)$ as $\vec{l}$.
Corollary 4.1. Assume that $k_k \geq k_{k-1} \geq \cdots \geq k_d \geq 10$. Any learning algorithm for $M_k$ makes at least the following number of mistakes in the worst case, regardless of the query sequence.  

$$
\frac{1}{2^d} \min\left\{ \left\lfloor \log k_1 \right\rfloor, k_d \right\} \prod_{j=2}^{d} \left\lfloor \log k_j \right\rfloor + \frac{1}{2} \sum_{j=2}^{d} (n_j - k_j) \left\lfloor \log k_j \right\rfloor \\
+ \frac{1}{2} (n_1 - k_1) \min\left\{ \left\lfloor \log k_1 \right\rfloor, k_d \right\}
$$

Proof. First we define a function $F_0 : \mathcal{R}^d \to \mathcal{R}$ by $F_0(\mathbf{v}) = \prod_{j=1}^{d} v_j + \sum_{j=1}^{d} v_j \times (n_{j+1} - k_{j+1})$, where $\mathbf{v} = (v_1, \ldots, v_d)$. Define $\mathbf{I} \in \mathcal{R}^d$ as follows.

$$
l_d = \frac{1}{2} \min\left\{ \left\lfloor \log k_1 \right\rfloor, k_d \right\}
$$

and

$$
l_j = \frac{1}{2} \left\lfloor \log k_{j+1} \right\rfloor \\
\text{for } j = 1, \ldots, d - 1.
$$

Now let $\mathbf{I}' = (I'_1, \ldots, I'_d) = (l_1, \ldots, l_d)$. Then under the condition $k_j \geq 10$ for all $j \in [d]$,

$$
l'_{j+1} + 2l'_{j+1} \leq \frac{1}{2} \log(2k_{j+2}) + \sqrt{2k_{j+1}} \leq k_{j+1}
$$

holds for every $j \in \{d, 1, 2, \ldots, d - 2\}$, and

$$
l'_d + 2l'_d \leq \left\lfloor \frac{1}{2} k_d \right\rfloor + \sqrt{2k_d} \leq k_d
$$

holds. Therefore, by Theorem 4.1, any algorithm makes at least $F(\mathbf{I}')$ mistakes, and the corollary follows because $F_0(\mathbf{I}) \leq F_0(\mathbf{I}') \leq F(\mathbf{I})$. \qed

4.1.2. An Upper Bound for Self-directed Learning. SD-predict described in Section 3.1 for learning binary relations can be extended to a learning algorithm for
learning general \(d\)-ary relations. We call the extension for \(d\)-ary relations SD-predict\((d)\) (See Fig. 9.) Let \(M \in \mathcal{M}_k\) be the target matrix. Note that various definitions in Section 3.1 are extended as in Definitions in Fig. 9.

Note that partition \(P_1^w \times \cdots \times P_d^w\) must be consistent with \(M_w\) just before Phase 1 of the algorithm.

**Theorem 4.2.** For all \(M \in \mathcal{M}_k\), SD-predict\((d)\) makes at most \(\prod_{j=1}^d k_j + \sum_{j=1}^d (n_j - k_j)\log k_j\) mistakes.

**Proof.** This theorem can be proved in the same way as Theorem 3.1, so we only sketch the proof. First, we define \(j\)-cost for each \(S_j(\bar{i})\) as the number of mistakes made by SD-predict\((d)\) in the process for \(S_j(\bar{i})\) minus the number of true partitions which has nonempty intersection with \(S_j(\bar{i})\) and have no known entry at the beginning of the process for \(S_j(\bar{i})\). With this notion of \(j\)-cost, we can show (via Lemmas 4.1 and 3.2) that the following holds for each \(j \in [d]\).

\[(\ast)\] The total sum (throughout a learning session) of \(j\)-cost is at most \((n_j - k_j)\log k_j\).

Since the total number of mistakes made by SD-predict\((d)\) is at most the summation of all \(j\)-costs plus the number of true partitions, the theorem follows from \((\ast)\).

The following lemma can be proved similarly to Lemma 3.1, and the proof is omitted.

**Lemma 4.1.** The \(j\)-cost for any \(S_j(\bar{i})\) is at most \(\min\{u, 2u_0 + 1, 2u_1 + 1\} - h\), provided that SD-predict\((d)\) has made mistakes predicting the entries in \(S_j(\bar{i})\), where \(u_0\) and \(u_1\) are the number of 0-entries and 1-entries, respectively, in \(S_j(\bar{i})\) with respect to true matrix \(M\), and \(h\) is the number of true partitions which has nonempty intersection with \(S_j(\bar{i})\).

By the same argument as that in the proof of Theorem 3.1, we see that the total \(j\)-cost can be bounded above by the total sum of \(j\)-costs for all \(S_j(\bar{i})\) that are divided by SD-predict\((d)\). The set of the pairs of the number of elements and the number of containing true partitions for all work partitions in \(P_d^w\) are initially \(\{([n_j], k_j)\}\) and finally \(\{([I], 1): I \in P_j^M\}\), where \(P_j^M\) denotes the true partition of \([n_j]\) induced by the \(j\)-submatrix types. Whenever \(S_j(\bar{i})\) is divided, the corresponding number pair \((u, h)\) is divided into \((u_0, h_0)\) and \((u_1, h_1)\) with the \(j\)-cost of at most \(\min\{u, 2u_0 + 1, 2u_1 + 1\} - h\). Thus, the total \(j\)-cost is bounded above by the sum of these \(j\)-costs throughout the process starting from \(\{([n_j], k_j)\}\) and ending at \(\{([I], 1): I \in P_j^M\}\).

Lemma 3.2 bounds this total \(j\)-cost by \((n_j - k_j)\log k_j\).

Note that the coefficient on \(n_j\) in the upper bound of Theorem 4.2 is only twice the coefficient on \(n_j\) in the lower bound shown in Corollary 4.1 for each \(j\).
Algorithm SD-predict(d)

Definitions:

\(M_w\) : Observation matrix (updated after each trial)

\(P_w^1 \times \ldots \times P_d^w\) : Work partitions

\(I_j(i)\) : Unique member of \(P_j^w\) containing \(i\)

\(S_j(i_1, \ldots, i_d) = \{(a_1, \ldots, a_d) : a_j \in I_j(i_j), (\forall h \neq j) a_h = i_h\}\)

Initialization: \(P_w^1 \times \ldots \times P_d^w = \{[n_1]\} \times \ldots \times \{[n_d]\}\).

Repeat the following 3 phases until all entries are known:

1. Pick an unknown entry \(\vec{i} = (i_1, \ldots, i_d)\) in the current observation matrix \(M_w\).

2. If \(M_w(S_1(\vec{i}))\) contains both 0 and 1, then divide \(I_1(i_1)\) into \(\{ j \in I_1(i_1) : M_w(j, i_2, \ldots, i_d) = 0 \}\) and \(\{ j \in I_1(i_1) : M_w(j, i_2, \ldots, i_d) = 1 \}\) and update the work partition \(P_w^1\) accordingly.

3. For all \(h \in [d]\) such that \(M_w(S_h(i'_1, i_2, \ldots, i_d))\) contains both 0 and 1, predict all unknown entries in \(S_h(i'_1, i_2, \ldots, i_d)\) by majority vote and divide \(I_h(i_h)\) if necessary as in Phase 2.

If mistakes were made, let \((i'_1, i_2, \ldots, i_d)\) denote the entry at which the first mistake was made.

FIG. 9. Self-directed learning algorithm for \(d\)-ary relations.
4.2. Teacher-Directed Learning

In this section, we analyze mistake bounds in the teacher-directed model, where a helpful teacher selects the trial sequence, and the evaluation of prediction performance is done by the worst case mistake bound over all concepts in the target class and all consistent learners, whose hypothesis is consistent with all the known instances. In this model, we can prove an upper bound and a lower bound in a way similar to the analogous proofs for $k$-binary relation in [4].

**Theorem 4.3.** The number of mistakes made by any consistent learner in teacher-directed learning is at most $\prod_{j=1}^{d} k_j + \sum_{j=1}^{d} (k_j - 1)(n_j - k_j)$.

**Proof.** Without loss of generality, we can assume that $j$-submatrices at all $a \in [k_j]$ are different from each other for any $j = 1, \ldots, d$. First, the teacher presents the learner all the entries in $D = \{(i_1, \ldots, i_d) : i_j \leq k_j\}$. Then, the teacher lets the learner know the type of $j$-submatrix at $h$ for all $j = 1, \ldots, d$ and all $h = k_j + 1, \ldots, n_j$ by presenting $k_j - 1$ entries in $\{(i_1, \ldots, i_d) : i_j = h, (\forall j' \neq j) i_{j'} < k_j\}$ which distinguish the $j$-submatrix at $h$ from the other $k_j - 1$ types. Thus, once these $\prod_{j=1}^{d} k_j + \sum_{j=1}^{d} (k_j - 1)(n_j - k_j)$ entries have been shown, the learner will possess the knowledge of the whole matrix and never makes a mistake from then on. □

Note that, in the case of $(k, l)$-binary relations with $k \neq l$, the upper bound of $\min\{k m + (k - 1)(n - k), ln + (l - 1)(m - l)\}$ that can be derived from Theorem 3 in [4] is at most the above upper bound, which is $kl + (k - 1)(n - k) + (l - 1)(m - l)$ in this case.

**Theorem 4.4.** The number of mistakes made by a consistent learner in teacher-directed learning is at least $k^d + (k - 1) \sum_{j=1}^{d} (n_j - k)$ in the worst case, where $k = \min\{k_1, \ldots, k_d\}$.

**Proof.** Let $M$ be the target matrix whose value $M(i_1, \ldots, i_d)$ is 1 only when $2 \leq i_1 = i_2 = \cdots = i_d \leq k$. Define $D$ as $D = \{(i_1, \ldots, i_d) : i_j \leq k\}$. In order to inform the learner all the types present in the target matrix, the teacher must show all entries in $D$. To let the learner know that the type of $j$-submatrix at $h$ is the type of $j$-submatrix at 1 for each $j = 1, \ldots, d$ and each $h = k + 1, \ldots, n_j$, the teacher must present all $k - 1$ entries in $\{(i_1, \ldots, i_d) : i_j = h, (\forall j' \neq j) i_{j'} = i, 2 \leq i \leq k\}$. Thus, in the worst case, a consistent learner can make at least $k^d + (k - 1) \sum_{j=1}^{d} (n_j - k)$ mistakes in total. □

4.3. Adversary-Directed Learning

4.3.1. A Lower Bound for Adversary-Directed Learning.

**Theorem 4.5.** Assume\(^{11}\) that $k_1 \geq k_2 \geq \cdots \geq k_d$. Any algorithm for \(\mathcal{A}_k\) makes at least

$$\prod_{j=1}^{d} k_j + (n_1 - k_1) \min\{\log k_1, k_2\} + \sum_{j=2}^{d} (n_j - k_j) \log k_j$$

mistakes in the worst case in the adversary-directed model.

\(^{11}\)Again, this assumption can be removed if $k_1, k_d$, and $n_1$ in the bound are replaced with $k_{\max}, k_{\min}$, and $n_{\max}$, respectively.
Proof. First the adversary selects all entries in $D_d = \{(i_1, \ldots, i_d) : i_j \in \lceil \log k_d \rceil, i_d \in [n_d], (\forall j \neq 1, d) i_j = 1\}$ and returns the opposite values to the learner’s predictions. In the 2-dimensional submatrix $D_d$, call the first dimension row and the $d$th dimension column. Then we can show that a subset $J_d \subseteq [n_d]$ containing 1 and having size $\min \{\lceil \log k_1 \rceil, k_d\}$ can be chosen such that (i) any two rows $a_1$ and $a_2$ that are distinct remain distinct even when the $d$th dimension is restricted to $J_d$; (ii) the set of column vectors $(b \in J_d)$ are either all distinct or exhaust all column vectors $(b \in [n_d])$. Such $J_d$ can be chosen as follows. Let $k$ be the number of distinct rows. Note that $k \leq \lceil \log k_d \rceil \leq \min \{\lceil \log k_1 \rceil, k_d\}$. Then, we can choose a set of at most $k$ distinct columns such that all distinct rows remain distinct even when restricted to that set of the columns. Next, if there is a column in this set which is identical to the column 1 vector, then replace that column with column 1. Otherwise, add column 1 to the set. Now it is easy to see that the set satisfies (i) and (ii), and that (i) and (ii) will remain satisfied, when we add as many distinct columns as possible to this set. Next the adversary selects all entries in $D_1 = \{(i_1, \ldots, i_d) : i_j \in \lceil \log k_d \rceil + 1, n_1, i_d \in J_d, (\forall j \neq 1, d) i_j = 1\}$ and returns the opposite values to the learner’s predictions. It is easy to see that, after predicting the entries in $D_d$ and $D_1$, the number of $d$-submatrix types is at most $k_d$ since every $d$-submatrix is of the same type as one of the $d$-submatrices at some $i_d \in J_d$. Similarly, the number of 1-submatrix types is at most $k_1$, since distinct 1-submatrices remain distinct when the $d$th dimension is restricted to $J_d$. Note that condition (i) is necessary for restricting the number of 1-submatrix types, and condition (ii) is necessary for restricting the number of $d$-submatrix types.

Then the adversary repeats the following procedure for $j = 1, \ldots, d - 2$ (cf. Fig. 10).

1. Focusing on the submatrix $D_j$ with the $j$th dimension being the column, construct a subset $J_j \subseteq [n_j]$ containing 1 of size $\lceil \log k_{j+1} \rceil$ such that the set of column vectors $(b \in J_j)$ is either all distinct or exhausts all column vectors $(b \in [n_j])$.

2. Select all remaining entries in $D_{j+1} = \{(i_1, \ldots, i_d) : i_j \in J_j, i_{j+1} \in [n_{j+1}], (\forall h \neq j, j + 1)i_h = 1\}$ and return the opposite values to the learner’s predictions.

After predicting all the $D_j$, the number of $j$-submatrix types is at most $k_j$ for all $j$. We can show this fact as follows. As for $d$-submatrices, $i_d = 1$ for all $D_j$ but $D_d$ and

FIG. 10. A query sequence in the proof of Theorem 4.2.
and the number of $d$-submatrix types is determined solely by the entry values of $D_d$ and $D_1$, which have been shown to be at most $k_d$ above. As for 1-submatrices, $i_j = 1$ for all $D_j$ but $D_d, D_1$ and $D_2$, and 1-submatrices are classified into at most $k_1$ types by the values of $D_d$ and $D_1$ as shown above. The entry values of $D_2$ will not make the number of 1-submatrix types exceed $k_1$, because the number of elements in $J_1$ is $\lfloor \log k_2 \rfloor$, which is at most $k_1$, and $J_1$ is chosen so as to contain as many distinct 1-submatrices as possible. As for $j$-submatrices, $1 < j < d - 1, i_j = 1$ for all $D_j$ but $D_j$ and $D_{j+1}$, and $j$-submatrices are classified into at most $k_j$ types by the values of $D_j$ because the number of elements in $J_{j-1}$ is $\lfloor \log k_j \rfloor$. The entry values of $D_{j+1}$ do not make the number of $j$-submatrix types exceed $k_j$ for the same reason as before. As for $(d - 1)$-submatrices, $i_{d-1} = 1$ for all $D_j$ but for $D_{d-1}$, and the number of $(d - 1)$-submatrix types can be seen to be at most $k_{d-1}$ similarly.

After the above procedure, we can construct a partition $P_1 \times \cdots \times P_d$ of $[n_1] \times \cdots \times [n_d]$ satisfying $|P_j| = k_j$ for all $j \in [d]$ and consistent with the known entries so far. From then on, the adversary returns the opposite value to the learner’s prediction for the first entry to be predicted in each $x \in P_1 \times \cdots \times P_d$ and returns consistently for the other entries.

By counting the total number of mistakes in this case, we can show the bound in the theorem.

Since the cardinality of $\mathcal{M}$ is at most $2^\prod_{j=1}^d k_j \prod_{j=1}^d k_j^n$, the halving algorithm [7], which is not a polynomial time algorithm for the current learning problem, makes at most $\prod_{j=1}^d k_j + \sum_{j=1}^d n_j \log k_j$ mistakes. So, the lower bound shown in Theorem 4.5 is quite tight.

4.3.2. The Cross Methods for Adversary-Directed Learning. In this section, we analyze the cross extensions of two existing algorithms for learning $k$-binary relations; ConsMajorityPredict [4] and Learn-Relation [6] (which we call Cross-ConsMajorityPredict and Cross-Learn-Relation, respectively). The term cross extension is derived from their 2-dimensional extensions which, when predicting entry $(i,j)$, not only make use of the known entries in column $j$, but also use the known entries in row $i$. Extending them for the general $d$-dimensional case, they use all the known entries $x'$ such that $x'$ is different from $x$ in just one coordinate, when predicting entry $x$.

We first describe Cross-ConsMajorityPredict. For each dimension $j \in [d]$, the algorithm keeps an edges $e_j(x,y)$ between the $j$-submatrix at $x$ and the $j$-submatrix at $y$ for all pairs $x, y \in [n_j]$. Each edge $e$ has a weight $w(e)$ associated with it, which is initially 1. Assume that the next entry to predict is $(i_1, \ldots, i_d)$. For $a = 0, 1$, let $E_a$ denote the set of edges $e_j(x, i_j)$ for all $j \in [d]$ and all $x \in [n_j]$ such that entry $(i_1, \ldots, i_{j-1}, x, i_{j+1}, \ldots, i_d)$ is known to assume the value $a$. Define $W_a$ to be the sum of the weights of edges in $E_a$. The algorithm predicts 1 when $W_1 \geq W_0$, and predicts 0 otherwise. When the algorithm’s prediction is $a$ and it is revealed to be wrong, all the weights in $E_a$ are set to 0. The algorithm repeats the prediction and the weight update described above for each trial. (See the first algorithm in Fig. 11.)
Algorithm Cross-ConsMajorityPredict

Perform the following five steps for each trial:

1. Receive an entry \((i_1, ..., i_d)\) to be predicted

2. Calculate \(W_a\) for \(a = 0, 1\) as follows:

\[
E_a = \bigcup_{j \in [d]} \{ e_j(x, i_j) : x \in [n_j], M_w(i_1, ..., i_{j-1}, x, i_{j+1}, ..., i_d) = a \}
\]

\[
W_a = \sum_{e \in E_a} w(e)
\]

3. If \(W_1 > W_0\) then predict with 1
   
   If \(W_1 < W_0\) then predict with 0
   
   Otherwise predict 0 or 1 at random

4. Receive correct label \(M(i_1, ..., i_d)\) and let \(M_w(i_1, ..., i_d) = M(i_1, ..., i_d)\)

5. If the current prediction \(a\) was wrong

   Let \(w(e) = 0\) for all \(e \in E_a\)

Algorithm Cross-Learn-Relation

Perform the following five steps for each trial:

1-4. The same steps as Cross-ConsMajorityPredict

5. If the current prediction \(a\) was wrong

   Let \(w(e) = 0\) for all \(e \in E_a\)

   Let \(w(e) = 2w(e)\) for all \(e \in \bar{E_a}\)

**FIG. 11.** Adversary-directed learning algorithms for \(\mathcal{A}_k\).
the algorithm predicts by weighted majority, the amount of weight decrease is larger
than the amount of weight increase at each weight update. Note that Cross-Learn-
Relation becomes noise-tolerant as Learn-Relation [6] by modifying the way of
weight update as multiplying $\omega(e)$ by $\gamma$ for $e \in E_a$ and by $(2 - \gamma)$ for $e \in E_\bar{a}$ at Step 5.
We can obtain an upper bound on the number of mistakes for this noise-tolerant
version of Cross-Learning-Relation from a modification of the proof of noise-free
version. We mention this in the last part of this section.

We now show upper bounds on the number of mistakes made by these algorithms.
The force of a mistake plays an important role in our proof as in [6]. Let $B(z_1, \ldots, z_d)$
for $z_1 \in [k_1], \ldots, z_d \in [k_d]$ be a submatrix of $M$ composed of entries $(i_1, \ldots, i_d)$ such
that $j$-submatrix at $i_j$ is of type $z_j$ for all $j \in [d]$. When the algorithms make
a wrong prediction for an entry $\bar{x} \in B(z_1, \ldots, z_d)$, the force of the mistake is defined
to be the number of known entries $\bar{x}'$ in $B(z_1, \ldots, z_d)$ such that $\bar{x}'$ is different
from $\bar{x}$ in just one coordinate. Let $F(z_1, \ldots, z_d)$ denote the sum of the forces of all
mistakes made when predicting an entry in $B(z_1, \ldots, z_d)$. Let $F_{\text{total}}$ denote
$\sum_{i_1=1}^{k_1} \cdots \sum_{i_d=1}^{k_d} F(z_1, \ldots, z_d)$.

The following lemma (Lemma 4.2) is a generalization of Lemma 4 in [6] and lower
bounds the force as a function of the number of mistakes when learning a general
d-ary relation.

**Proposition 4.1** (in the proof of Lemma 4 in [6]). Let $\langle x \rangle^n$ denote a sequence
containing the symbol $x$ repeated $n$ times. Let $\sigma$ denote the sum of the first $\mu$ elements
of the sequence $\langle 0 \rangle^n \langle 1 \rangle^n \langle 2 \rangle^n \cdots$. Then, $\sigma \leq \mu^2/(2n) - \mu/2$.

**Lemma 4.2.** Let $r(z_1, \ldots, z_d)$ be the number of mistakes made when predicting
entries in $B(z_1, \ldots, z_d)$. Then, for any $(z_1, \ldots, z_d)$, it holds that

$$F(z_1, \ldots, z_d) \geq \frac{1}{2} \sum_{j=1}^{d} \left( \frac{r^2(z_1, \ldots, z_d) n_{j,z_j} - r(z_1, \ldots, z_d)}{\prod_{i=1}^{d} n_{i,z_i}} \right),$$

where $n_{j,z_j}$ is the number of $j$-submatrix of type $z_j$.

**Proof.** Suppose that the $r$th mistake in $B(z_1, \ldots, z_d)$ is made when predicting
entry $(i_1, \ldots, i_d)$. Let $I_j^r$ denote the set of known entries such that all but the $j$th
coordinate are the same as the entry $(i_1, \ldots, i_d)$. Let $\sigma_j^r$ denote the number of
entries in $I_j \cap B(z_1, \ldots, z_d)$. Then, $F(z_1, \ldots, z_d) = \sum_{j=1}^{d} \sum_{r(z_1, \ldots, z_d)} \sigma_j^r$. For each $j$,
$\sum_{r(z_1, \ldots, z_d)} \sigma_j^r$ is at least the sum of the first $r(z_1, \ldots, z_d)$ elements of the sequence
$\langle 0 \rangle \prod_{i=1}^{d} n_{i,z_i}/n_{z_j} \langle 1 \rangle \prod_{i=1}^{d} n_{i,z_i}/n_{z_j} \langle 2 \rangle \prod_{i=1}^{d} n_{i,z_i}/n_{z_j} \cdots$. By Proposition 4.1, this is bounded
from below by $r^2(z_1, \ldots, z_d) n_{j,z_j}/(2 \prod_{i=1}^{d} n_{i,z_i}) - r(z_1, \ldots, z_d)/2$. Thus, $F(z_1, \ldots, z_d)$ is
at least $(1/2) \sum_{j=1}^{d} (r^2(z_1, \ldots, z_d) n_{j,z_j}/\prod_{i=1}^{d} n_{i,z_i} - r(z_1, \ldots, z_d))$. □

**Proposition 4.2.** Let $h$ be a nonnegative function on $[k]$. Let $g$ be a positive
function on $[k]$ such that $\sum_{x=1}^{k} g(x) = \sigma$. Then, $\sum_{x=1}^{k} h(x)/g(x) \geq (\sum_{x=1}^{k} \sqrt{h(x)})^2/\sigma$.

**Proof.** The proof is straightforward by applying the method of Lagrange
multipliers. □
**Proposition 4.3.** Let $f$ be a nonnegative function on $[k_1] \times \cdots \times [k_d]$ such that $\sum_{x_1=1}^{k_1} \cdots \sum_{x_d=1}^{k_d} f(x_1, \ldots, x_d) = \mu$. For each $i \in [d]$, let $g_i$ be a positive function on $[k_i]$ such that $\sum_{x_i=1}^{k_i} g_i(x_i) = \sigma_i$. Then,

$$\sum_{x_1=1}^{k_1} \cdots \sum_{x_d=1}^{k_d} f^2(x_1, \ldots, x_d) \prod_{i=1}^{d} g_i(x_i) \geq \frac{\mu^2}{\prod_{i=1}^{d} \sigma_i}.$$

**Proof.** For $i \in [0, d]$, define function $h_i(x_{i+1}, \ldots, x_d)$ as follows:

$$h_0(x_1, \ldots, x_d) = f^2(x_1, \ldots, x_d)$$
$$h_i(x_{i+1}, \ldots, x_d) = \sum_{x_i=1}^{k_i} \frac{h_{i-1}(x_1, \ldots, x_d)}{g_i(x_i)} \quad \text{for } i \geq 1.$$ 

Then, we have

$$h_d() = \sum_{x_1=1}^{k_1} \cdots \sum_{x_d=1}^{k_d} \frac{f^2(x_1, \ldots, x_d)}{\prod_{i=1}^{d} g_i(x_i)}.$$

By applying Proposition 4.2 repeatedly on each summation, we obtain

$$h_d() \geq \frac{1}{\sigma_d} \left( \sum_{x_1=1}^{k_1} \sqrt{h_{d-1}(x_{d})} \right)^2$$
$$\geq \frac{1}{\sigma_d} \left( \sum_{x_1=1}^{k_1} \sqrt{\frac{1}{\sigma_{d-1}} \left( \sum_{x_{d-1}=1}^{k_{d-1}} \sqrt{h_{d-2}(x_{d-1}, x_d)} \right)^2} \right)^2$$
$$= \frac{1}{\sigma_d \sigma_{d-1}} \left( \sum_{x_{d-1}=1}^{k_{d-1}} \sum_{x_1=1}^{k_1} \sqrt{h_{d-2}(x_{d-1}, x_d)} \right)^2$$
$$\vdots$$
$$\geq \frac{1}{\prod_{i=1}^{d} \sigma_i} \left( \sum_{x_1=1}^{k_1} \cdots \sum_{x_{d-1}=1}^{k_{d-1}} \sqrt{h_0(x_1, \ldots, x_d)} \right)^2$$
$$= \frac{\mu^2}{\prod_{i=1}^{d} \sigma_i}. \quad \blacksquare$$

**Lemma 4.3.** Let $\mu$ be the number of mistakes made by either of the two algorithms (Cross-ConsMajorityPredict and Cross-Learn-Relation). Then,

$$\mu \leq f(\bar{n}, \bar{k})d + \sqrt{2f(\bar{n}, \bar{k})F_{\text{total}}}$$

holds, where $f(\bar{n}, \bar{k}) = \prod_{i=1}^{d} n_i / \sum_{i=1}^{d} (n_i / k_i)$. 
Proof. Let \( r(z_1, \ldots, z_d) \) denote the number of mistakes made when predicting entries in \( B(z_1, \ldots, z_d) \). By Lemma 4.2,

\[
\frac{1}{2} \sum_{j=1}^{d} \sum_{z_1=1}^{k_1} \cdots \sum_{z_d=1}^{k_d} \left( \frac{r^2(z_1, \ldots, z_d)n_{j_1 \cdots j_d} - r(z_1, \ldots, z_d)}{\prod_{i=1}^{d} n_{i,z_i}} \right) \leq F_{total} \quad (4)
\]

holds. Let \( \mu_{j,z} = \sum_{z_1=1}^{k_1} \cdots \sum_{z_{j-1}=1}^{k_{j-1}} \sum_{z_{j+1}=1}^{k_{j+1}} \cdots \sum_{z_d=1}^{k_d} r(z_1, \ldots, z_d) \). Then,

\[
\sum_{z_1=1}^{k_1} \cdots \sum_{z_d=1}^{k_d} \frac{r^2(z_1, \ldots, z_d)n_{j_1 \cdots j_d}}{\prod_{i=1}^{d} n_{i,z_i}} \geq \sum_{z_1=1}^{k_1} \sum_{j=1}^{d} \frac{\mu_{j,z}^2 n_j}{\prod_{i=1}^{d} n_i} \geq \frac{\mu^2}{\prod_{i=1}^{d} n_i} \sum_{j=1}^{d} n_j - d \mu \quad (5)
\]

holds. Note that the first inequality can be seen to hold from Proposition 4.3, and the second inequality can be shown by the method of Lagrange multipliers. Thus,

\[
\frac{1}{2} \left( \sum_{j=1}^{d} \frac{n_j}{\prod_{i=1}^{d} n_i} \mu^2 - d \mu \right) \leq F_{total}.
\]

Therefore,

\[
\mu \leq \frac{\prod_{i=1}^{d} n_i}{\sum_{i=1}^{d} \frac{n_i}{m_i}} d + \sqrt{2 \prod_{i=1}^{d} n_i F_{total}}.
\]

Let \( \mathcal{E} \) denote the set of all edges, and define \( \mathcal{E}_0 \) as follows:

\( \mathcal{E}_0 = \bigcup_{j=1}^{d} \{e_j(x,y)\} \): the \( j \)-submatrix at \( x \) and the \( j \)-submatrix at \( y \) are of the same type.

**Lemma 4.4.** Assume that \( n_i > k_i \) for all \( i \in [d] \). Then,

\[
F_{total} \leq \frac{1}{2} \sum_{i=1}^{d} \frac{k_i^2 - 1}{k_i^2} n_i(n_i - 1)
\]

during the learning session of Cross-ConsMajorityPredict.

**Proof.** Let \( \mathcal{E}_1 = \mathcal{E} - \mathcal{E}_0 \). Note that only the weights of edges in \( \mathcal{E}_1 \) decrease during the learning session, and the weight of an edge in \( \mathcal{E}_0 \) is always 1. Since the algorithm predicts by weighted majority, the total weight decreases by at least \( F_{total} \) during the course of a learning session. Thus, by using the method of Lagrange
multipliers and the assumption that \( n_i > k_i \) for all \( i \in [d] \), we have
\[
F_{\text{total}} \leq |\mathcal{E}| - |\mathcal{E}_0| \\
= \sum_{i=1}^{d} \left( \frac{1}{2} n_i(n_i - 1) - \frac{1}{k_i} n_i n_i - 1 \right) \\
\leq \frac{1}{2} \sum_{i=1}^{d} \left( n_i(n_i - 1) - \frac{1}{k_i} n_i - k_i \right) \\
\leq \frac{1}{2} \sum_{i=1}^{d} \left( n_i(n_i - 1) - \frac{1}{k_i} n_i - 1 \right) \\
\leq \frac{1}{2} \sum_{i=1}^{d} \frac{k_i^2 - 1}{k_i} n_i(n_i - 1). \quad \blacksquare
\]

**Lemma 4.5.** During the learning session of Cross-Learn-Relation,
\[
F_{\text{total}} \leq \frac{\log e}{2e} \sum_{i=1}^{d} n_i(n_i - 1)
\]
holds.

**Proof.** First, note that \( 2^{\log e} = \prod_{e \in \mathcal{E}_0} w(e) \). Thus, by the concavity of the log function,
\[
F_{\text{total}} = \sum_{e \in \mathcal{E}_0} \log w(e) \leq |\mathcal{E}_0| \log \left( \frac{\sum_{e \in \mathcal{E}_0} w(e)}{|\mathcal{E}_0|} \right) \\
\leq |\mathcal{E}_0| \log \left( \frac{\mathcal{E}}{|\mathcal{E}_0|} \right) \quad (6)
\]
holds. Since \( f(x) = x \ln (a/x) \leq f(a/e) = a/e \), \( F_{\text{total}} \leq \frac{\log e}{2e} |\mathcal{E}|. \quad \blacksquare \)

**Theorem 4.6.** Assume that \( n_i > k_i \) for all \( i \in [d] \). Then, algorithm Cross-ConsMajorityPredict makes at most
\[
f(\mathbf{n}, \mathbf{k}) d + \sqrt{f(\mathbf{n}, \mathbf{k}) \sum_{i=1}^{d} \frac{k_i^2 - 1}{k_i} n_i(n_i - 1)}
\]
mistakes on an adversary-selected trial sequence, where \( f(\mathbf{n}, \mathbf{k}) = \prod_{i=1}^{d} n_i / \sum_{i=1}^{d} (n_i/k_i) \).

**Proof.** This result immediately follows from Lemmas 4.3 and 4.4. \( \blacksquare \)

**Theorem 4.7.** Algorithm Cross-Learn-Relation makes at most
\[
f(\mathbf{n}, \mathbf{k}) d + \sqrt{f(\mathbf{n}, \mathbf{k}) \log e \sum_{i=1}^{d} n_i(n_i - 1)}
\]
mistakes on an adversary-selected trial sequence, where \( f(\mathbf{n}, \mathbf{k}) = \prod_{i=1}^{d} n_i / \sum_{i=1}^{d} (n_i/k_i) \).
Proof. This result immediately follows from Lemmas 4.3 and 4.5. ■

Note that $\log e/e \leq (k^2 - 1)/k^2$ when $k \geq 2$.

We say that matrix $M$ is balanced if, for each $j \in [d]$, the number of $j$-submatrices of type $z_j$ is just $n_j/k_j$ for all $z_j \in [k_j]$.

**Corollary 4.2.** Assume that $n_i > k_i$ for all $i \in [d]$. Then, when the target matrix is balanced, algorithm Cross-Learn-Relation makes at most

$$f(\mathbf{n}, \mathbf{k})d + \sqrt{2f(\mathbf{n}, \mathbf{k}) \sum_{i=1}^{d} \frac{\log(\max_i k_i)}{k_i} n_i(n_i - 1)}$$

mistakes on an adversary-selected query sequence, where $f(\mathbf{n}, \mathbf{k}) = \prod_{i=1}^{d} n_i/\sum_{i=1}^{d} (n_i/k_i)$.

Proof. By Inequality (6) in the proof of Lemma 4.5, $F_{\text{total}} \leq |\mathcal{E}_0|\log|\mathcal{E}|/|\mathcal{E}_0|$ holds. Since the target matrix is balanced, $|\mathcal{E}_0| = 1/2 \sum_{i=1}^{d} n_i(n_i - k_i)/k_i$ holds. Thus, we have

$$\frac{1}{2} \sum_{i=1}^{d} \frac{n_i(n_i - 1)}{k_i} \geq |\mathcal{E}_0| \geq \frac{1}{2} \sum_{i=1}^{d} \frac{n_i(n_i - 1)}{k_i^2} \geq \frac{1}{2} \sum_{i=1}^{d} \frac{n_i(n_i - 1)}{2(\max_i k_i)^2}.$$  

Using both the upper and lower bounds on $|\mathcal{E}_0|$, we have

$$F_{\text{total}} \leq \frac{1}{2} \sum_{i=1}^{d} \frac{n_i(n_i - 1)}{k_i} \log(\max_i k_i)^2 = \sum_{i=1}^{d} \frac{\log(\max_i k_i)}{k_i} n_i(n_i - 1).$$

Plugging in the above in Lemma 4.3 completes the proof. ■

We define a cross method as an algorithm which predicts the label of an entry $\mathbf{x}$ as a function of only the values of the known entries $\mathbf{x}'$ such that $\mathbf{x}'$ differs from $\mathbf{x}$ in just one coordinate. Both Cross-ConsMajorityPredict and Cross-Learn-Relation are clearly cross methods.

Note that the upper bounds given in Theorems 4.6 and 4.7 both approach $O(n^{d-1})$, when the $n_i$ are roughly the same (and equal $n$) and much larger than the $k_j$, and $d \geq 3$. This is a rather disappointing bound, considering that the size of the entire domain is $O(n^d)$. Furthermore, for each $j \in [d]$, if we regard a $j$-submatrix as a row, we can directly apply ConsMajorityPredict [4] and Learn-Relation in [6] to learn a $d$-ary relation and obtain mistake bounds of $k_j \prod_{i=1}^{d} n_i/n_j + n_j \sqrt{(k_j - 1) \prod_{i=1}^{d} n_i/n_j}$ and $k_j \prod_{i=1}^{d} n_i/n_j + n_j \sqrt{3(\prod_{i=1}^{d} n_i/n_j) \log k_j}$, respectively. The first term of these bounds is $k_j \prod_{i=1}^{d} n_i/n_j$ and the first term of the bounds for their cross-method extensions is $d \prod_{i=1}^{d} n_i/\sum_{i=1}^{d} (n_i/k_i)$. Note that, for every bound considered up to now, the first term dominates the bound when $d$ is large and all $n_i$ are almost the same. Since $\min k_j \prod_{i=1}^{d} n_i/n_j \leq d \prod_{i=1}^{d} n_i/\sum_{i=1}^{d} (n_i/k_i) \leq \max k_j \prod_{i=1}^{d} n_i/n_j$, the bounds obtained by the cross extensions fall between the bounds of the original algorithm for the best $j$ and the worst $j$.
The next theorem shows that these limitations are in fact inherent in any deterministic cross methods, indicating that in order to dramatically improve these bounds, we would need to design learning algorithms that use more information than the observed data whose indices differ from the current tuple in just one coordinate.

**Theorem 4.8.** Any deterministic cross method makes at least \( (\min_n n_i)^{d-1} \) mistakes in the adversary-directed model, either for the constant 1 function or the constant 0 function, represented by matrices \( M^0 \) and \( M^1 \), respectively.

**Proof.** Let \( m = \min_n n_i \). By Lemma 4.6, we can choose a set \( S \) of \( n_i^{d-1} \) entries such that any two different entries in \( S \) differ in more than one coordinates. The adversary selects all the entries in \( S \) first. Any deterministic cross method must predict all these entries with the same value, because there are no observed entries that its prediction depends upon. Thus, the algorithm makes mistakes for all the entries in \( S \) either for the target matrix \( M^0 \) or \( M^1 \). \( \blacksquare \)

Recall that the non pure relations are those relations having a specified number of types in each dimension, disregarding a number of exceptions. Here we show how Cross-Learn-Relation can be extended for learning such relations. The noise-tolerant version of Cross-Learn-Relation multiplies \( w(e) \) by \( \gamma \) for \( e \in E_d \) and by \( (2 - \gamma) \) for \( e \in E_d \) at Step 5. In order to obtain an upper bound on the number of mistakes made by this algorithm, we can use Lemma 4.3 without modification, but must modify Lemma 4.5 so as to take noise into account. Let \( \delta(z_1, \ldots, z_d) \) denote the number of noisy entries in \( B(z_1, \ldots, z_d) \), namely, \( \delta(z_1, \ldots, z_d) = \min\{u_0, u_1\} \) where \( u_0 \) and \( u_1 \) are the number of 0-entries and 1-entries in \( B(z_1, \ldots, z_d) \), respectively. We define \( J \) to be the number of times that a weight in \( \delta_0 \) is multiplied by \( \gamma \). Then, by the same argument as in Lemma 2 in [6], we can obtain

\[
F_{\text{total}} \leq \frac{b}{a} J + \frac{\log e}{2ea} \sum_{i=1}^{d} n_i(n_i - 1),
\]
where \( a = \log(2 - \gamma) \) and \( b = \log(2 - \frac{\gamma}{2}) \). Since the quantity \( J \) is increased by at most \( \sum_{j=1}^{d} n_{jz_i} \) for each noise in \( B(z_1, \ldots, z_d) \), the inequality

\[
J \leq \sum_{z_1=1}^{k_1} \cdots \sum_{z_d=1}^{k_d} \delta(z_1, \ldots, z_d) \sum_{j=1}^{d} n_{jz_i} \\
\leq \sum_{j=1}^{d} n_j \sum_{z_1=1}^{k_1} \cdots \sum_{z_d=1}^{k_d} \delta(z_1, \ldots, z_d) \tag{8}
\]

holds. Let \( z \) denote the total number of noisy entries, namely, \( z = \sum_{z_1=1}^{k_1} \cdots \sum_{z_d=1}^{k_d} \delta(z_1, \ldots, z_d) \). Then, by Lemma 4.3, Inequality (7), and Inequality (8), we obtain the following mistake bound for nonpure \( d \)-ary relations:

\[
f(\mathbf{\bar{t}}, \mathbf{\bar{k}}) d + \sqrt{\frac{b z}{a} \sum_{i=1}^{d} n_i + f(\mathbf{\bar{t}}, \mathbf{\bar{k}}) \log e \sum_{i=1}^{d} n_i (n_i - 1)}.
\]

4.3.3. A Hardness Result for Designing Efficient Algorithms with a Nearly Optimal Mistake Bound. Designing efficient algorithms with a nearly optimal mistake bound for a \( d \)-ary relation (\( d \geq 3 \)) in the adversary-directed model appears to be challenging even if the number of \( j \)-submatrix types is at most two for each \( j \in [d] \). In this section we show that the sample consistency problem for \( \mathcal{M}(2,2,2) \) is NP-complete, although there is a polynomial time algorithm solving the same problem for \( \mathcal{M}(2,2) \), supporting the above intuition.

We define the sample consistency problem for any class \( \mathcal{M} \) of \( d \)-dimensional matrices as follows: Given a \( d \)-dimensional observation matrix \( M_p \) that is partially known, decide if there is \( M \in \mathcal{M} \) that is consistent with \( M_p \).

**Theorem 4.9.** There is a polynomial time algorithm that solves the sample consistency problem for \( \mathcal{M}(2,2) \).

**Proof.** We use the same notation as in Section 3.2. Let \( M_p : [n] \times [m] \rightarrow \{0, 1, \ast\} \) be the input partially known 2-dimensional matrix. We can decide if there is a L-type matrix \( M \) consistent with \( M_p \) by checking the consistency of two partitions \( \{R_x, [n]\}, \{C_x, [m]\}, \{x = 0, 1\} \) of \([n] \times [m] \), where \( R_x \) is the set of rows containing \( x \) and \( C_x \) is the set of columns containing \( x \). So, to complete the proof, we need only show that the existence of a non-L-type matrix consistent with \( M_p \) can be decided. Construct graph \( G_r \) for \( M_p \) as defined in Section 3.2. Let \( G'_r \) be the subgraph of \( G_r \) which is composed of the vertices corresponding to the representative rows and the undirected edges indicating a type disagreement. We show that the existence of a non-L-type matrix consistent with \( M_p \) is equivalent to the existence of a 2-coloring of the graph \( G'_r \), whose decidability is known to be solvable in polynomial time.

Assume the existence of a partition \( \{P_{r_0}, P_{r_1}\} \times \{P_{r_0}, P_{r_1}\} \) of \([n] \times [m] \) consistent with \( M_p \). Assign one color to vertices of \( G'_r \) in \( P_{r_0} \) and the other color to vertices of \( G'_r \) in \( P_{r_1} \). Then, the consistency of the partition ensures that every pair of vertices of \( G'_r \) connected by an edge has different colors.

Assume the existence of a 2-coloring of the graph \( G'_r \). The coloring can be extended to one for the graph \( G_r \) by coloring each vertex \( i \) with the same color as the representative vertex in \( G'_r \), that is, the vertex directed by an edge going out from \( i \).
Consider the partition \( \{ P_{00}, P_{11} \} \) of \([n]\) defined by this coloring of \(G_r\). Then for every column \(j\), all the known rows in one of the two partitions must have value 1, and value 0 in the other, for otherwise it would not be a 2-coloring. Let \(P_{i0} = \{ j \in [m]: \exists i \in P_{00} \text{ s.t. } M_p(i,j) = 0 \} \cup \{ j \in [m]: \exists i \in P_{11} \text{ s.t. } M_p(i,j) = 1 \}\). It is easy to see that Partition \( \{ P_{00}, P_{11} \} \) of \([n]\) is consistent with \(M_p\).

Next, we show that the sample consistency problem for \(M_{(2,2,2)}\) is NP-complete by reducing the NP-complete 3-Set-Splitting [2] described below to this problem.

3-Set-Splitting.

**INSTANCE:** A collection \(S = \{ s_1, \ldots, s_m \}\) of subsets \(s_i \subseteq [n]\) each with \(|s_i| = 3\).

**QUESTION:** Is there a partition of \([n]\) into two subsets \(T_0\) and \(T_1\) such that no \(s_i \in S\) is entirely contained in either \(T_0\) or \(T_1\)?

**Theorem 4.10.** The sample consistency problem for \(M_{(2,2,2)}\) is NP-complete.

**Proof.** It is easy to see that the sample consistency problem is in NP, so it remains to show that it is NP-hard. Let \(S = \{(a_1, b_1, c_1), \ldots, (a_m, b_m, c_m)\}\) be an instance of 3-Set-Splitting. We will construct a partially known \((n+m) \times 3m \times 2m\) matrix \(M_p\) such that there exists \(M \in M_{(2,2,2)}\) consistent with \(M_p\) if and only if \(S\) is a positive instance.

In \(M_p\), the values of entries \((a_j, 3j-2, 1), (b_j, 3j-1, 1), (c_j, 3j, 1), (n+j, 3j-2, m+j),\) and \((n+j, 3j, m+j)\) are known to be 1, 0, 1, 0, respectively, for each \(j \in [m]\) (see Fig. 12). The values of all other entries are assumed to be unknown.

Suppose that there is a matrix \(M \in M_{(2,2,2)}\) that is consistent with \(M_p\). Let \(T_0\) be the set of 1-submatrices of the same type as the 1-submatrix at 1, and let \(T_1\) be the set of 1-submatrices of the other type. Now suppose that there is a triple \((a_j, b_j, c_j) \in S\) such that \(\{a_j, b_j, c_j\} \subseteq T_0\) or \(\{a_j, b_j, c_j\} \subseteq T_1\); i.e., 1-submatrix types of \(a_j, b_j\) and \(c_j\) are all the same. In that case, it follows that the 2-submatrix types of \(3j-2, 3j-1,\) and

![FIG. 12. Transforming \((a_j, b_j, c_j)\) to five known entries.](image-url)
3j must be all distinct which contradicts the assumption that $M \in \mathcal{M}_{(2,2,2)}$. (In particular, it follows that $(a_j, 3j-2, j), (b_j, 3j-2, j), (c_j, 3j-2, j), (n+j, 3j-2, m+j)$ are 1, 1, 1, 1, $(a_j, 3j-1, j), (b_j, 3j-1, j), (c_j, 3j-1, j)$ are 0, 0, 0, and $(a_j, 3j, j), (b_j, 3j, j), (c_j, 3j, j), (n+j, 3j, m+j)$ are 1, 1, 1, 0.) Thus, there is no $(a_j, b_j, c_j) \in S$ entirely contained in either $T_0$ or $T_1$.

Next, assume that there is a partition $\{T_0, T_1\}$ of $[n]$ such that no $(a_j, b_j, c_j) \in S$ is entirely contained in either $T_0$ or $T_1$. We construct a matrix $M \in \mathcal{M}_{(2,2,2)}$ consistent with $M_p$. In other words, we construct a partition $\{U_{e0}, U_{e1}\}$ of the set of $e$-submatrices for each $e \in [3]$ such that the values of all known entries in each of $\{U_{10}, U_{11}\} \times \{U_{20}, U_{21}\} \times \{U_{30}, U_{31}\}$ are identical. Define partition $\{U_{e0}, U_{e1}\}$ for each $e \in [3]$ as follows:

1. $i \in U_{1x}$ if $i \in T_x$
2. $3j-2 \in U_{2x}, 3j-1 \in U_{2x}, n+j \in U_{1x}, j, m+j \in U_{3x}$ if $\{a_j, b_j\} \subseteq T_x$
3. $3j \in U_{2x}, 3j-2 \in U_{2x}, n+j \in U_{1x}, j \in U_{3x}, m+j \in U_{3x}$ if $\{b_j, c_j\} \subseteq T_x$

where $x = 0$ or 1. We then define matrix $M$ by specifying its entry values for the eight partitions as in Fig. 13.

It is not difficult to verify that $M$ as defined above is consistent with $M_p$. (Figure 14 gives example 3-submatrices for three of the cases described above.)

![FIG. 13. Consistent entry values for partitions $\{U_{10}, U_{11}\} \times \{U_{20}, U_{21}\} \times \{U_{30}, U_{31}\}$.](image1)

![FIG. 14. Schematic views of 3-submatrix partitions for three cases: (1) $x = 0$, (2) $x = 0$, and (3) $x = 1$.](image2)
5. CONCLUDING REMARKS

For many practical applications, algorithms in the adversary-model with a good performance are desired. The theoretical results presented in this paper seem to indicate that designing learning algorithms with a good performance guarantee for the adversary-directed model is challenging. In particular, one of our results regarding the cross methods indicates that to hope for a better performance guarantee, we must design learning algorithms that go beyond the cross methods, that is, one that make use of observed data whose indices differ in many coordinates from the current tuple to be predicted. In that case, one must resolve the so-called credit assignment problem, namely of deciding which coordinate was to blame for a prediction mistake. Although some progress has been made on this problem experimentally [1], no theoretical results to date have shown that the general $d$-ary relations are efficiently learnable with a reasonable mistake bound. An interesting open problem is to find such an algorithm.

ACKNOWLEDGMENTS

We thank Dr. K. Nakamura, Mr. T. Fujita, Dr. S. Doi, and Dr. S. Goto of NEC Corporation for their encouragement.

REFERENCES

8. Deleted in proof.