Machine Learning, 20, 245–271 (1995)
© 1995 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Learning Binary Relations Using Weighted Majority Voting *

SALLY A. GOLDMAN sg@cs.wustl.edu
Dept. of Computer Science, Washington University, St. Louis, MO 63130

MANFRED K. WARMUTH manfred@cs.ucsc.edu
Dept. of Computer and Information Sciences, University of California, Santa Cruz, CA 95064

Editor: David Haussler

Abstract. In this paper we demonstrate how weighted majority voting with multiplicative weight updating can be applied to obtain robust algorithms for learning binary relations. We first present an algorithm that obtains a nearly optimal mistake bound but at the expense of using exponential computation to make each prediction. However, the time complexity of our algorithm is significantly reduced from that of previously known algorithms that have comparable mistake bounds. The second algorithm we present is a polynomial time algorithm with a non-optimal mistake bound. Again the mistake bound of our second algorithm is significantly better than previous bounds proven for polynomial time algorithms.

A key contribution of our work is that we define a "non-pure" or noisy binary relation and then by exploiting the robustness of weighted majority voting with respect to noise, we show that both of our algorithms can learn non-pure relations. These provide the first algorithms that can learn non-pure binary relations.

Keywords: on-line learning, mistake-bounded learning, weighted majority voting, noise tolerance, binary relation

1. Introduction

In this paper we demonstrate how weighted majority voting with multiplicative weight updating can be applied to obtain robust algorithms for learning binary relations. Following Goldman, Rivest and Schapire (1993), a binary relation is defined between two sets of objects, one of cardinality n and the other of cardinality m. For all possible pairings of objects, there is a predicate relating the two sets of variables that is either true (1) or false (0). The relation is represented as an $n \times m$ matrix M of bits, whose (r, j) entry is 1 if and only if the relation holds between the corresponding elements of the two sets. Furthermore, there are a limited number of object types. Namely, the matrix M is restricted to have at most k distinct row types among its n rows. (Two rows are of the same type if they agree in all columns.) This restriction is satisfied whenever there are only k types of identical objects in the set of n objects being considered in the relation.

We study the problem of learning binary relations under the standard on-line (or incremental) learning model (Littlestone 1989; Littlestone 1988). The *learning session* consists of a sequence of *trials*. In each trial, the learner must predict the value of some

* The first author was supported in part by NSF grant CCR-91110108 and NSF National Young Investigator Grant CCR-9357707 with matching funds provided by Xerox Corporation, Palo Alto Research Center and WUTA. The second author was supported by ONR grant NO0014-91-J-1162 and NSF grant IRI-9123692.

unknown matrix entry that has been selected by the adversary[1]. After predicting, the learner receives the value of the matrix entry in question as *feedback*. If the prediction disagrees with the feedback, then we say the learner has made a *mistake*. The learning session continues until the learner has predicted each matrix entry. The goal of the learner is to make as few prediction mistakes as possible.

One could view each row as a point in $\{0,1\}^m$ and each row type of a binary relation as a cluster of points in $\{0,1\}^m$ that are distance zero from each other. Initially, the learner knows nothing about how the points may be clustered, yet as the learning session proceeds the learner must infer both how the points are clustered (since this will be essential in reducing the mistakes) and the value of the m bits defining each cluster. We thus use the term *cluster* rather than row type when talking about the grouping of the rows.

One contribution of this paper is a formal definition for the problem of learning binary relations in which the clusters are not "pure" in that two rows in the same cluster may not be exactly the same. In other words, we remove the requirement that all points in a cluster are distance zero from each other. Thus even if the target matrix were known there remains an interesting clustering problem that one could address, namely, what clustering achieves the optimal tradeoff between the number of clusters and the distances between the points in the clusters. Of course, our learning algorithm must perform the clustering in an on-line manner while also predicting the values of unknown matrix entries. By viewing the problem of learning binary relations under this more general formulation, we achieve two important goals. First, such a modification in the definition of the learning problem much better matches the motivation provided by Goldman, Rivest, and Schapire (1993). Second, we can greatly reduce the mistake bounds obtained under the original formulation of Goldman et al. since now a group of rows that have slight differences can be viewed as a single cluster rather than having each row as its own cluster. In addition, the final weights of the second algorithm we present may be used to obtain a solution to the problem of how to cluster the rows of the target matrix.

A second contribution of this paper is two new algorithms (both based on weighted majority voting with multiplicative weight updating) to learn both pure and non-pure binary relations. Weighted majority voting provides a simple and effective method for constructing a learning algorithm A that is provided with a pool of "experts", one of which is known to perform well, but A does not know which one. Associated with each expert is a weight that gives A's confidence in the accuracy of that expert. When asked to make a prediction, A predicts by combining the votes from its experts based on their associated weights. When an expert suggests the wrong prediction, A passes that information to the given expert and reduces its associated weight. In this work, we use a multiplicative weight updating scheme to adjust the weights. Namely, the weight associated with each expert that mispredicts is multiplied by some weight $0 \leq \beta < 1$. By selecting $\beta > 0$ this algorithm is robust against noise in the data. This robust nature of weighted majority voting enables us to apply our algorithms to the problem of learning non-pure relations by viewing discrepancies between the rows in the same cluster as noise.

When using weighted majority voting there is a tradeoff between the number of experts (i.e. the number of weights being combined) and the computation time needed to make each prediction. Our first algorithm applies a weighted majority scheme with $k^n/k!$ weights to the problem of learning binary relations. We show that the mistake bound of this algorithm nearly matches the information theoretic lower bound. While the computation time needed to make each prediction is exponential, it is still significantly reduced from that used by the halving algorithm (Barzdin & Freivald, 1972; Littlestone 1988; Angluin 1988) [2]. In our second algorithm we apply a weighted majority voting scheme that uses only a polynomial number of weights and thus can make each prediction in polynomial time. While the mistake bound proven for this algorithm is not near optimal, with respect to what can be obtained with unlimited computation, it is significantly better than previous bounds obtained by polynomial time algorithms. We first present and analyze our two algorithms for the problem of learning pure binary relations. We then show how to exploit the robustness of weighted majority voting, to generalize both of these algorithms to obtain the first known algorithms for learning non-pure binary relations.

Our second algorithm has two novel features that required the use of a new method for analyzing the mistake bound. This algorithm runs n copies of a weighted majority voting scheme in parallel where in each trial only one of the n copies is active. However, all n copies share weights and thus cooperatively work to learn the target relation. Another unusual property of our voting scheme is that all experts begin with incomplete information and thus do *not* always know how to make each prediction. Furthermore, the prediction an expert would make for any given matrix entry can change over time. (In fact, initially no experts will know how to vote for any matrix entry.) As the learning session proceeds all experts monotonically gain information and thus know how to vote more often. Thus to bound the number of mistakes made by our second algorithm, we must determine the minimum rate at which information is being gained across the n copies of our voting scheme.

The remainder of this paper is organized as follows. In the next section, we discuss the halving algorithm and the particular weighted majority voting scheme, WMG, that we apply. In Section 3 we present our two different algorithms for applying WMG to the problem of learning a binary relation. In Section 4 we formally define the problem of learning non-pure binary relations, and demonstrate how the robust nature of WMG can be exploited to handle such noise. In Section 5 we present our main result. Namely, we provide a technique that enables us to prove an upper bound on the number of mistakes made by our polynomial-time algorithm for learning non-pure binary relations. Finally, in Section 6 we close with some concluding remarks.

2. Preliminaries

In the noise-free setting, if the learner has unlimited computation time then the halving algorithm (Barzdin & Freivald, 1972; Littlestone 1988; Angluin 1988) typically performs very well. The halving algorithm predicts according to the majority of the relations that are consistent with the completed trials, and thus each mistake halves the number of remaining relations. Since the number of binary relations is at most $2^{km}k^n/k!$ the standard halving algorithm makes at most

$$\begin{aligned}\lg(2^{km}k^n/k!) &= km + n\lg k - \lg k! \\ &\leq km + n\lg k - k\lg(k/e) \\ &= km + (n-k)\lg k + k\lg e\end{aligned}$$

mistakes [3]. The halving algorithm can be viewed as keeping $2^{km}k^n/k!$ weights, one weight per possible binary relation. Initially, all weights start at 1, and whenever a binary relation becomes inconsistent with the current partial matrix, its weight is set to 0. To make a prediction for a given matrix entry, each binary relation votes according to its bit in that entry. Since the halving algorithm predicts according to the majority of consistent relations (i.e. those with weight 1), each mistake halves the total weight in the system. Since the initial weight is $2^{km}k^n/k!$ and the final weight is at least 1, at most $km + (n-k)\lg k + k\lg e$ mistakes can occur.

Observe that the time used to make each prediction is linear in the number of weights. Thus we are interested in algorithms that use a small number of weights in representing their hypotheses. The algorithms we present update the weights according to a variant of the weighted majority algorithm of Littlestone and Warmuth (1989) called WMG. We view WMG as a node that is connected to each expert (or input) by a weighted edge. The inputs are in the interval $[0,1]$. An input x of weight w votes with weight xw for 1 and with weight $(1-x)w$ for 0. The node "combines" the votes of the inputs by determining the total weight q_0 (respectively q_1) placed on 0 (respectively 1) and predicts with the bit corresponding to the larger of the two totals (and for the sake of discreteness with 1 in case of a tie). After receiving feedback of what the prediction should have been, for each input, the fraction of the weight placed on the wrong bit is multiplied by β, where $\beta \in [0,1)$. Thus the weight w of an input x becomes $(1-x+x\beta)w$ if the feedback is 0 and $((1-x)\beta + x)w$ if the feedback is 1. If $\beta = 0$ then the total weight halves in each trial in which the node makes a mistake and we obtain an analysis like that of the halving algorithm.

3. Our Algorithms For Applying WMG

In this section we describe two different methods for applying WMG to the problem of learning a binary relation. The first algorithm uses one node and $k^n/k!$ weights. Thus the number of weights is still exponential but lower than the number of weights used by the halving algorithm by a multiplicative factor of 2^{km}. For this case, the analysis is straightforward, and for the noise-free case, the bounds achieved are nearly optimal with respect to the known information-theoretic lower bound. The purpose of this algorithm is to show what is possible when computational resources are cheap. In the second algorithm we use one node for each of the n rows and one weight for each pair of rows (i.e. $\binom{n}{2}$ weights). Proving bounds for the second algorithm is much more involved and is the focus of the paper. The bounds obtained are non-optimal when compared to those obtained when computation time is not a concern. However, our bounds are significantly better than previous bounds obtained by a polynomial algorithm.

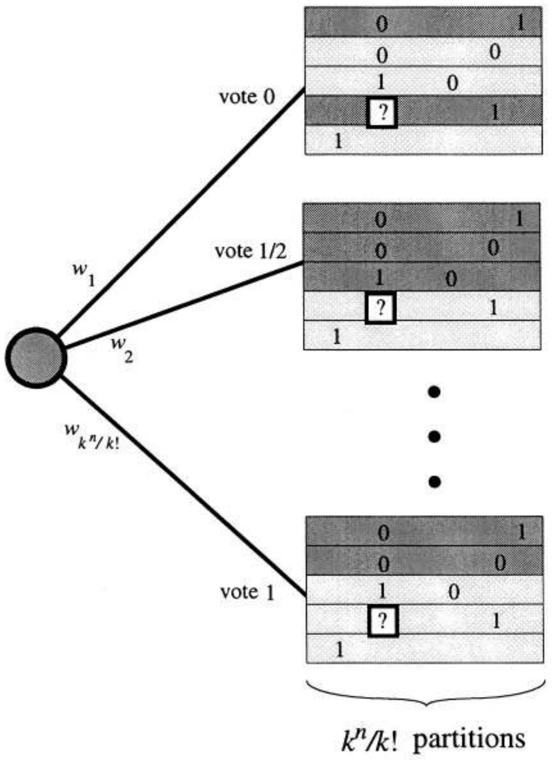

Figure 1. This figure illustrates how the voting works in our first algorithm. In this example $k = 2$. We use the two degrees of shading to indicate how the rows of the matrices are partitioned. Thus on the right we have shown the partially known matrix under three different partitions. Just to the left of each partition we show its vote for the unknown matrix entry. Recall that a partition voting with $1/2$ can be viewed as a vote of 1 with half of its weight and a vote of 0 with half of its weight. So the overall prediction made in this example is 1 if and only if $w_2/2 + \cdots + w_{k^n/k!} \geq w_1 + w_2/2 + \cdots$.

We now describe our two algorithms in more detail. In the first algorithm we use one weight per partition of the n rows into at most k clusters (i.e. $k^n/k!$ weights). Initially all weights are set to 1. To make a prediction for a new matrix entry, each partition votes as follows: If a column of the cluster to which the new entry belongs has already been set then vote with the value of this bit; otherwise, vote with 1/2 causing the weight to be split between the votes of 0 and 1. Our algorithm predicts according to the weighted majority of these votes (see Figure 1). Recall that after receiving the feedback WMG multiplies the fractions of the weights that were placed on the wrong bit by β. By selecting $\beta = 0$, the weight of partitions that predict incorrectly (and are thus inconsistent with the partial matrix) are set to zero and the weights of all partitions that split their vote are halved.

After all entries of the target matrix are known, the correct partition has weight at least 2^{-km} since it never predicted incorrectly, and split its vote at most km times. Since the initial weight is $k^n/k!$, we obtain the mistake bound of $km + (n-k)\lg k + k \lg e$ just as for the halving algorithm. (Actually, one can show that when $\beta = 0$ then the first algorithm simulates the halving algorithm with $k^n/k!$ weights instead of $2^{km}k^n/k!$ weights). Note that the mistake bound of this algorithm is essentially optimal since Goldman, Rivest, and Schapire (1993) prove an information-theoretic lower bound of $km + (n-k)\lfloor \lg k \rfloor$ mistakes. While this algorithm has assumed that an upper bound on k is known, if no such bound is provided the standard doubling trick can be applied. Namely, the learner begins with an estimate, $\hat{k} = 1$, as an upper bound for k. If the algorithm fails (i.e. makes too many mistakes) then $\hat{k}$ is doubled and the process in repeated. Once $\hat{k} \geq k$ the algorithm will succeed. Observe that the final value $\hat{k}_f$ of $\hat{k}$ will be less than $2k$. Also, since $\hat{k}$ is growing at an exponential rate, the computation time and mistakes made during the iteration with $\hat{k} = \hat{k}_f$ dominates the other runs. More specifically the number of mistakes made by this variation for k unknown is at most

$$\begin{aligned}
&\sum_{i=0}^{\lg \hat{k}_f} (2^i m + ni - i2^i + 2^i \lg e) \\
&= 2\hat{k}_f m + \frac{n}{2} \lg \hat{k}_f (\lg \hat{k}_f + 1)(2\hat{k}_f(\lg \hat{k}_f - 1) + 2) + 2\hat{k}_f \lg e \\
&< 2\hat{k}_f m + \frac{n}{2} (\lg \hat{k}_f + 1)^2 - 2\hat{k}_f (\lg \hat{k}_f - 1) + 2\hat{k}_f \lg e \\
&\leq 4km + \frac{n}{2} (\lg k + 2)^2 - 2k \lg k + 4k(\lg e + 1) \\
&= O(km + n \lg^2 k - k \lg k) = O(km + (n \lg k - k) \lg k)
\end{aligned}$$

where the first step uses the equality $\sum_{i=0}^{n} i2^i = 2^{n+1}(n-1) + 2$.

In our second algorithm we use one weighted majority node per row of the matrix, and one edge between each pair of nodes. Thus, unlike the first algorithm, no knowledge of k is needed. Let $e_{rr'}$ (and $e_{r'r}$) denote the (undirected) edge between the node for row r and the node for row r', and let $w(e)$ denote the weight of edge e. The node for row r dictates the predictions for all entries in row r. So the nodes, in some sense, partition

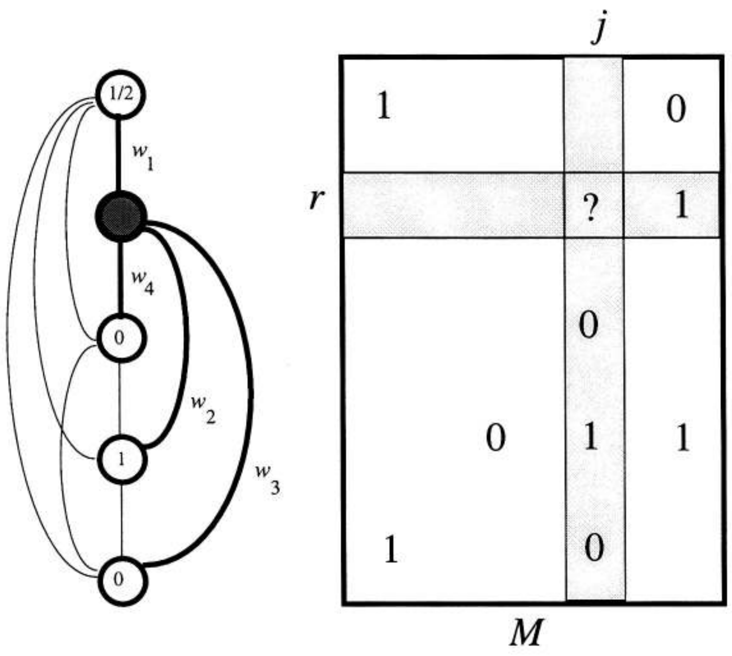

Figure 2. This figure illustrates how predictions are made in our second algorithm. The weighted majority node corresponding to row r (heavily shaded) is used to predict M_{rj}. The predictions of the inputs coming from the nodes are shown. So for this example the shaded node predicts 1 if $w_1/2 + w_2 \geq w_1/2 + w_3 + w_4$ and predicts 0 otherwise.

the learning problem among themselves. Assume M_{rj} is the next value to predict. To make its prediction, the node for row r combines the votes of the $n-1$ inputs from column j of matrix M. Each node $r' \neq r$ votes with the weight $w(e_{rr'})$ for the bit at $M_{r'j}$. If the bit $M_{r'j}$ is unknown (corresponding to its vote being $1/2$) then the weight is split between the two votes (see Figure 2). If a mistake occurs then only the $n-1$ weights connected to node r are adjusted using a multiplicative weight update scheme[4]. Thus we are running n copies of WMG in parallel, where in each trial only one of the copies makes a prediction. We show that this parallel composition of weighted majority style algorithm "learns" in the sense that if there are few clusters in the matrix then the algorithms eventually cooperate to make few mistake altogether. This holds even if the adversary gets to choose in what order the entries of the matrix M are uncovered[5]. Also note that the final weight between two nodes can be used to partition the rows into clusters so that one obtains a good tradeoff between the number of clusters and the distances within each cluster. Namely, clusters should be formed of rows for which the weight between all pairs is sufficiently high.

In Section 5 we provide a technique to adapt the worst-case mistake bounds proven for the weighted majority algorithm WMG to this parallel application. As a corollary to our main theorem, we show that when $\beta = 0$, our second algorithm obtains a mistake bound of $km + \min\left\{\frac{n^2}{2e}\lg e, n\sqrt{3m\lg k}\right\}$ using only $\binom{n}{2}$ weights. Here k is the size of the smallest partition consistent with the whole matrix. The best previous bound for a polynomial algorithm was $km + n\sqrt{(k-1)m}$ (Goldman, Rivest & Schapire, 1993). An interesting aspect of our problem, besides its parallel nature, is that when node r is to predict for entry M_{rj}, not all other $n-1$ entries in the j-th column may have been uncovered. Such unknown ("sleeping") entries are naturally set to $1/2$, leading to split votes.

There are many relatives of the basic weighted majority algorithm that we could use within our two algorithms such as a probabilistic variant due to Vovk (1990) (see also Cesa-Bianchi, Freund, Helmbold, Haussler, Schapire and Warmuth (1993)). Also the Vee algorithm of Kivinen and Warmuth (1994) handles inputs and predictions in [0,1] and the algorithm of Littlestone, Long and Warmuth (1991) has small square loss against the best convex combination of the inputs. (Incidentally all these on-line prediction algorithms have multiplicative weight updates in common.) We chose the simplest setting for showing the usefulness of our method: the entries of the matrix are binary and the predictions must be deterministic.

4. A Generalization: Non-Pure Relations

A key contribution of this paper is showing how the robust nature of WMG enables us to solve an important generalization of the problem of learning binary relations with noisy data. To motivate this problem we briefly review the allergist example given by Goldman, Rivest, and Schapire (1993). 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

```
**Learn-Relation**$(0 \leq \gamma < 1)$

For all $r, r'$ such that $(r \neq r')$ initialize $w(e_{rr'}) = 1$
In each trial do the following four parts:
1. Receive a matrix entry $M_{rj}$ for prediction
2. Produce a prediction as follows
   For each row $r' \neq r$
      If $M_{r'j}$ is not known then row $r'$ predicts that $M_{rj} = 1/2$
      If $M_{r'j} = 1$ then row $r'$ predicts that $M_{rj} = 1$
      If $M_{r'j} = 0$ then row $r$ predicts that $M_{rj} = 0$
   Let $R_0$ be the set of all rows that predict $M_{rj} = 0$
   Let $R_1$ be the set of all rows that predict $M_{rj} = 1$
   Let $W_0 = \sum_{r' \in R_0} w(e_{rr'})$
   Let $W_1 = \sum_{r' \in R_1} w(e_{rr'})$
   If $W_1 \geq W_0$ predict 1
   Else predict 0
3. Receive correct value for $M_{rj}$
4. If the prediction of the algorithm was wrong then update the weights as follows
   For each $r' \neq r$
      If row $r'$ made a correct prediction then let $w(e_{rr'}) \leftarrow (2 - \gamma)w(e_{rr'})$
      Else if row $r'$ predicted incorrectly then let $w(e_{rr'}) \leftarrow \gamma w(e_{rr'})$
```

Figure 3. Our polynomial prediction algorithm for learning binary relations. Note that in the for loops of Steps 2 and 4, one could just look at rows for which $w(e_{rr'}) > 0$. Also we obtain the same mistake bound if the update in Step 4 is performed at each step regardless of whether a mistake occurred.

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. What options does the allergist have in testing a patient for a given allergen? He/she could just perform the intradermal test (option 0). Another option (option 1) is to perform an epicutaneous test, and if it is not conclusive, then perform an intradermal test. Which option is best? If the patient has no allergy or a mild allergy to the given allergen, then 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 option 1 is best, since the patient does not experience a bad reaction. The allergist's goal here is to minimize the number of prediction mistakes in choosing the option to test each patient for each allergen. Although Goldman et al. explore several possible methods for the selection of the presentation order, here we only consider the standard worst-case model in which an adversary determines the order in which the patient/allergen pairs are presented.

This example makes an assumption that is a clear oversimplification. Namely, they assume that there are a common set of "allergy types" that occur often and that most people fit into one of these allergy types. Thus the allergy types become the clusters of the matrix. However, while it is true that often people have very similar allergies, there are not really pure allergy types. In other words, it is unreasonable to assume that all rows in the same cluster are identical but rather they are just close to each other. Without this flexibility one may be required to have most patient's allergies correspond to a distinct allergy type. Henceforth, we refer to the original formulation of the problem of learning binary relations in which all clusters are "pure" as *learning pure relations.*

We propose the following generalization of the problem of learning binary relations that we refer to as *learning non-pure relations.* For any column c of bits, let $\mathcal{N}_0(c)$ be the number of zeros in c. Likewise, let $\mathcal{N}_1(c)$ be the number of ones in c. Suppose that the rows of the matrix are partitioned into a set of k clusters $p = \{S^1, \ldots, S^k\}$. Let S^i_j denote the jth column of the cluster (or submatrix) S^i. For each cluster we define a distance measure

$$d(S^i) = \sum_{j=1}^{m} \min\{\mathcal{N}_0(S^i_j), \mathcal{N}_1(S^i_j)\}$$

In other words, think of defining a *center* point for partition S^i by letting the value of column j in this center be the majority vote of the entries in S^i_j. Then $d(S^i)$ is just the sum over all rows s in S^i of the Hamming distance between s and this center point.

We define the *noise* of partition p, α_p, as

$$\alpha_p = \sum_{S^i \in p} d(S^i) = \sum_{S^i \in p} \sum_{j=1}^{m} \min\{\mathcal{N}_0(S^i_j), \mathcal{N}_1(S^i_j)\},$$

and the *size* of partition p, k_p, as the number of clusters in partition p. For each cluster i and column j, we define $\delta_{i,j} = \min\{\mathcal{N}_0(S^i_j), \mathcal{N}_1(S^i_j)\}$, and $\delta_i = \sum_{j=1}^{m} \delta_{i,j}$. Thus $\alpha_p = \sum_{i=1}^{k_p} \delta_i = \sum_{i=1}^{k_p} \sum_{j=1}^{m} \delta_{i,j}$. Due to the robust nature of WMG, we can use both algorithms to learn non-pure relations by simply using a non-zero update factor β.

We now discuss both of our algorithms when applied to the problem of learning non-pure relations and give bounds for each. The key to our approach is to view minor discrepancies between rows in the same cluster as noise. This greatly reduces the mistake bounds that one can obtain when using the original formulation of Goldman, Rivest, and Schapire (1993) by reducing the number of clusters. The robust nature of the weighted majority algorithm enables us to handle noise.

To demonstrate our basic approach, we now show that our first algorithm (i.e. the one using $k^n/k!$ weights) can learn a non-pure relation by making at most [6]

$$\min\left\{k_p m + \alpha_p + \frac{(n-k)\ln k + k + \alpha_p \ln \frac{1}{\beta}}{\ln \frac{2}{1+\beta}}\right\} \tag{1}$$

mistakes in the worst case, where $0 \le \beta < 1$ is the update factor, and the minimum is taken over all partitions p of size at most k and k_p denotes the size and α_p the noise of partition p.

In the noisy case the first algorithm still uses a single copy of the weighted majority algorithm with one weight for each of the $k^n/k!$ partitions. (See Figure 4 for the complete algorithm.) Assume M_{rj} is the next value to predict. Then a particular partition predicts with the majority of all already set entries from column j of rows in the same group as row r in the partition. In case of a tie the partition predicts with 1/2. (Note that in the noise-free setting all entries known in a given partition for a given column must be the same.) Finally, a partition with weight w and prediction x votes with xw for 1 and $(1-x)w$ for 0. The Algorithm WMG totals the votes for 0 and for 1 and predicts with the bit of the larger total (with 1 in case of a tie).

When a partition predicts incorrectly, its weight is multiplied by β. A partition that splits its vote has its weight multiplied by $(1+\beta)/2$. (Half of its weight remains unchanged and the other half is multiplied by β.) Finally, since WMG votes in agreement with at least half of the weight in the system, when a mistake occurs, at least half of the weight is multiplied by β (and the rest is unchanged). Thus for each trial in which a mistake occurs the total weight after the trial is at most $(1+\beta)/2$ times the total weight before the trial. We now argue that a partition p predicts incorrectly at most α_p times and splits its vote at most $k_p m + \alpha_p$ times. To see this consider what happens in column j of cluster i. The number of wrong predictions is at most $\delta_{i,j}$ (i.e. the number of minority bits) and the number of ties at most $1 + \delta_{i,j}$. Thus the number of times partition p predicts incorrectly is at most $\sum_{i=1}^{k_p}\sum_{j=1}^{m} \delta_{i,j} = \alpha_p$, and the number of times partition p splits its vote is at most $\sum_{i=1}^{k_p}\sum_{j=1}^{m}(1+\delta_{i,j}) = k_p m + \alpha_p$.

Thus it follows that the final weight in the system is at least $\beta^{\alpha_p}\left(\frac{1+\beta}{2}\right)^{k_p m + \alpha_p}$. Since the initial weight in the system is $k^n/k!$ and for each trial in which a mistake occurs the total weight after the trial is at most $(1+\beta)/2$ times the total weight before the trial, we get the following inequality for the total number of mistakes μ:

```
**Slow-Learn-Relation**$(0 \leq \beta < 1)$

For $1 \leq i \leq k^n/k!$, initialize $w_i = 1$
In each trial do the following four parts:
1. Receive a matrix entry $M_{rj}$ for prediction
2. Produce a prediction as follows
    For $1 \leq i \leq k^n/k!$
        Let $R$ be the set of rows in the same group as row $r$ under partition $i$
        Let $N_0$ be the number of rows from $R$ where column $j$ is known to be a 0
        Let $N_1$ be the number of rows from $R$ where column $j$ is known to be a 1
        If $N_1 = N_0$ then partition $i$ predicts that $M_{rj} = 1/2$
        If $N_1 > N_0$ then partition $i$ predicts that $M_{rj} = 1$
        Else partition $i$ predicts that $M_{rj} = 0$
    Let $R_0$ be the set of all partitions that predict $M_{rj} = 0$
    Let $R_1$ be the set of all partitions that predict $M_{rj} = 1$
    Let $W_0 = \sum_{i \in R_0} w_i + \sum_{i \in R-(R_0 \cup R_1)} w_i/2$
    Let $W_1 = \sum_{i \in R_1} w_i + \sum_{i \in R-(R_0 \cup R_1)} w_i/2$
    If $W_1 \geq W_0$ predict 1
    Else predict 0
3. Receive correct value for $M_{rj}$
4. If the prediction of the algorithm was wrong then update the weights as follows
    For $1 \leq i \leq k^n/k!$
        If partition $i$ made a prediction of $1/2$ then let $w_i \leftarrow w_i/2 + \beta \cdot w_i/2$
        If partition $i$ made an incorrect prediction (of 0 or 1) then let $w_i \leftarrow w_i \cdot \beta$
```

Figure 4. Our algorithm that uses $k^n/k!$ weights to obtain an nearly optimal algorithm for learning binary relations. We would obtain the same mistake bound if Step 4 is performed at each step regardless of whether a mistake occurred.

$$\frac{k^n}{k!}\left(\frac{1+\beta}{2}\right)^{\mu} \geq \beta^{\alpha_p}\left(\frac{1+\beta}{2}\right)^{k_p m+\alpha_p}$$

Solving for μ gives the bound given in Equation (1).

An interesting modification of our first algorithm would be to consider all rows in the same group as row r and then predict with the number of 1's already known in column j divided by the total number of known entries in column j (i.e. a prediction of $N_1/(N_0+N_1)$ using the notation of Figure 4). We did not use this rule because it is harder obtain a lower bound on the final weight in the system.

Finally, by applying the results of Cesa-Bianchi, Freund, Helmbold, Haussler, Schapire, and Warmuth (1993) we can tune β as a function of an upper bound α on the noise.

LEMMA 1 *(Cesa-Bianchi, Freund, Helmbold, Haussler, Schapire & Warmuth, 1993) For any real value $z \geq 0$ or $z = \infty$,*

$$\frac{z^2+\ln\frac{1}{g(z)}}{2\ln\frac{2}{1+g(z)}} \leq 1+z+\frac{z^2}{2\ln 2},$$

where $g(z) = \frac{1}{1+2z+\frac{z^2}{2\ln 2}}$ and $g(\infty) = 0$.

THEOREM 1 *For any positive integers k and α, the first algorithm with $\beta = g\left(\sqrt{\frac{(n-k)\ln k+k}{\alpha}}\right)$ makes at most*

$$\min\left\{k_p m + 3\alpha_p + 2\sqrt{\alpha((n-k)\ln k+k)} + (n-k)\lg k + k\lg e\right\}$$

mistakes, where the minimum is taken over all partitions p whose size k_p is at most k and whose noise α_p is at most α.

Proof: Since $\alpha_p \leq \alpha$ and $\left(\ln\frac{1}{\beta}\right)/\left(2\ln\frac{2}{1+\beta}\right) > 1$ for $\beta \in [0,1)$, it follows that

$$\begin{aligned}
&\frac{(n-k)\ln k+k+\alpha_p\ln\frac{1}{\beta}}{\ln\frac{2}{1+\beta}}\\
&= \frac{(n-k)\ln k+k}{\ln\frac{2}{1+\beta}} + 2\alpha_p + 2\alpha_p\left(\frac{\ln\frac{1}{\beta}}{2\ln\frac{2}{1+\beta}}-1\right)\\
&\leq \frac{(n-k)\ln k+k}{\ln\frac{2}{1+\beta}} + 2\alpha_p + 2\alpha\left(\frac{\ln\frac{1}{\beta}}{2\ln\frac{2}{1+\beta}}-1\right)\\
&= 2\alpha\left(\frac{z^2+\ln\frac{1}{\beta}}{2\ln\frac{2}{1+\beta}}\right) + 2\alpha_p - 2\alpha, \text{ where } z = \sqrt{\frac{(n-k)\ln k+k}{\alpha}}.
\end{aligned}$$

So by applying Lemma 1 with and $\beta = g(z)$ to the bound (1) we obtain a worst-case mistake bound of

$$\min\left\{k_p m + \alpha_p + 2\left(\alpha_p + \sqrt{\alpha\left((n-k)\ln k+k\right)} + \frac{(n-k)\ln k+k}{2\ln 2}\right)\right\}$$

$$= \min\left\{k_p m + 3\alpha_p + 2\sqrt{\alpha\left((n-k)\ln k + k\right)} + (n-k)\lg k + k\lg e\right\}$$

for our first algorithm, where the minimum is taken over all partitions p of size at most k with noise at most α, and k_p denotes the size and α_p the noise of partition p.

■

For the above tuning we needed an upper bound for both the size and the noise of the partition. If an upper bound for only one of the two is known, then the standard doubling trick can be used to guess the other. This causes only a slight increase in the mistake bound (see Cesa-Bianchi, Freund, Helmbold, Haussler, Schapire, and Warmuth (1993)). Note that in the above mistake bound there is a subtle tradeoff between the noise α_p and size k_p of a partition p.

Recall that when this first algorithm is applied in the noise-free case that it essentially matches the information-theoretic lower bound. An interesting question is whether or not it can be shown to be essentially optimal in the case of learning non-pure relations.

As we show in the next section (Theorem 3), when using the second algorithm for learning non-pure relations, our algorithm makes at most

$$min\left\{k_p m + \sqrt{3mn^2\lg k + 4\alpha_p mn\left(1 - \frac{\alpha_p}{mn}\right) + mn\sqrt{24\alpha n\left(1 - \frac{\alpha}{mn}\right)\ln k}}\right\}$$

mistakes in the worst case, where the minimum is taken over all partitions p, and k_p denotes the size and α_p the noise of partition p where $k_p \leq k$ and $\alpha_p \leq \alpha$.

5. Algorithm Two: A Polynomial-time Algorithm

In this section we analyze our second algorithm, *Learn-Relation*, when applied to learning non-pure relations. (Recall that *Learn-Relation* is shown in Figure 3.) The mistake bound of Theorem 2 obtained for this algorithm is larger than the mistake bound of the first algorithm. However this algorithm uses only $\binom{n}{2}$ weights as opposed to exponentially many.

We begin by giving an update that is equivalent to the one used in WMG (Littlestone & Warmuth, 1989). Recall that in WMG if x is the prediction of an input with weight w, then if the feedback is the bit ρ then w is multiplied by $1-(1-\beta)|x-\rho|$ for $\beta \in [0,1)$. If $\beta = \gamma/(2-\gamma)$, then for our application the update of WMG can be summarized as follows

- If a node predicts correctly (so, $|x-\rho| = 0$) its weight is not changed.
- If a node makes a prediction of $1/2$ then its weight is multiplied by $1/(2-\gamma)$.
- If a node predicts incorrectly (so, $|x-\rho| = 1$) then its weight is multiplied by $\gamma/(2-\gamma)$.

In the new update all factors in the above algorithm are simply multiplied by $(2 - \gamma)$. This update is used in our Algorithm *Learn-Relation(γ)* since it leads to simpler proofs. Because voting is performed by a weighted majority vote, the predictions made by the two schemes are identical. In order to use the analysis technique of Littlestone and Warmuth we must obtain a lower bound for the final weight in the system. However, using WMG the weight in the system is decreased by nodes that do not predict, and thus we would have to compute an upper bound on the total number of times that this occurs. Thus to simplify the analysis, we have modified the update scheme (i.e. at each step we multiplied all weights by $(2-\gamma)$) so that the weights of nodes that do not predict remain unchanged.

5.1. The Analysis

In this section we compute an upper bound on the number of mistakes made by *Learn-Relation.*

We begin with some preliminaries. Let $f_{xx} = \frac{\partial^2 f}{\partial x^2}$, $f_{yy} = \frac{\partial^2 f}{\partial y^2}$, and $f_{xy} = \frac{\partial^2 f}{\partial y \partial x}$. A function $f : \Re \rightarrow \Re$ is concave (respectively convex) over an interval D of $\Re$ if for all $x \in D$, $f_{xx}(x) \leq 0$ ($f_{xx}(x) \geq 0$). In our analysis we repeatedly use the following variants of Jensen's inequality. Let f be a function from $\Re$ to $\Re$ that is concave over some interval D of $\Re$. Let $q \in \mathcal{N}$, and let $x_1, x_2, \ldots, x_q \in D$. Then

$$\sum_{i=1}^{q} x_i = U \Rightarrow \sum_{i=1}^{q} f(x_i) \leq q f(U/q).$$

Furthermore, if f is monotonically increasing over the interval D then the following holds:

$$\sum_{i=1}^{q} x_i \leq U \Rightarrow \sum_{i=1}^{q} f(x_i) \leq q f(U/q).$$

Likewise, let f be a function from $\Re$ to $\Re$ that is convex over some interval D of $\Re$. Let $q \in \mathcal{N}$, and let $x_1, x_2, \ldots, x_q \in D$. Then

$$\sum_{i=1}^{q} x_i = U \Rightarrow \sum_{i=1}^{q} f(x_i) \geq q f(U/q).$$

We also use Jensen's inequality when applied to a function over two variables. A function $f : \Re \times \Re \rightarrow \Re$ is concave over an interval $D_x \times D_y$ of $\Re \times \Re$ if for all $x \in D_x$ and $y \in D_y$, $f_{xx} \leq 0$, $f_{yy} \leq 0$, and $f_{xx} f_{yy} - (f_{xy})^2 \geq 0$. Let f be a function from $\Re \times \Re$ to $\Re$ that is concave over some interval $D_x \times D_y$ of $\Re \times \Re$. Let $q \in \mathcal{N}$, and let $x_1, x_2, \ldots, x_q \in D_x$ and $y_1, y_2, \ldots, y_q \in D_y$. Then

$$\sum_{i=1}^{q} x_i = U_x \text{ and } \sum_{i=1}^{q} y_i = U_y \Rightarrow \sum_{i=1}^{q} f(x_i, y_i) \leq q f(U_x/q, U_y/q).$$

We now give an overview of the proof of our main result along with several key lemmas that are used in the proof. Let p be *any* partition of size k_p and noise α_p. We note that the partition p is fixed throughout the analysis. Furthermore quantities such as $\mathcal{E}$, F_i, δ_i (defined below) depend implicitly on p. Let μ denote the total number of mistakes made by the learner, and let μ_i denote the number of mistakes that occur when the learner is predicting an entry in a row of cluster i. (Thus, $\sum_{i=1}^{k_p} \mu_i = \mu$.) Let n_i be the number of rows in cluster i. (So $n = \sum_{i=1}^{k_p} n_i$.) Let $\mathcal{A}$ be all $\binom{n}{2}$ edges and the set $\mathcal{E}$ contain all edges connecting two rows of the same cluster. We further decompose $\mathcal{E}$ into $\mathcal{E}_1, \ldots, \mathcal{E}_{k_p}$ where $\mathcal{E}_i$ contains all edges connecting two rows in the same cluster i of p. Observe that $|\mathcal{E}_i| = \frac{n_i(n_i-1)}{2}$. When making an erroneous prediction for M_{rj}, we define the *force* of the mistake to be the number of rows in the same cluster as row r for which column j was known when the mistake occurred. Let F_i be the sum of the forces of all mistakes made when predicting an entry in a row of cluster i.

Recall that the noise of a partition p is defined as

$$\alpha_p = \sum_{S^i \in p} d(S^i) = \sum_{S^i \in p} \sum_{j=1}^{m} \min\{\mathcal{N}_0(S^i_j), \mathcal{N}_1(S^i_j)\} = \sum_{i=1}^{k_p} \sum_{j=1}^{m} \delta_{i,j} = \sum_{i=1}^{k_p} \delta_i.$$

We now define J_i to be the number of times that a weight in $\mathcal{E}_i$ is multiplied by γ when making a prediction for an entry in cluster i. That is, J_i is the total number of times, over all trials in the learning session in which a mistake occurs, where an entry in cluster i incorrectly predicts the value of an entry in cluster i (voting with all its weight).

We now give the key lemma used in our main proof. For ease of exposition, let $a = \lg(2-\gamma) = \lg \frac{2}{1+\beta}$ and $b = \lg\left(\frac{2-\gamma}{\gamma}\right) = \lg \frac{1}{\beta}$.

LEMMA 2 *For each* $1 \le i \le k_p$,

$$F_i = \frac{b}{a} J_i + \frac{1}{a} \sum_{e \in \mathcal{E}_i} \lg w(e).$$

Proof: We begin by noting that, if $\alpha_p = 0$, then $J_i = 0$ and the number of times some weight in $\mathcal{E}_i$ is multiplied by $(2-\gamma)$ equals F_i. Thus, in the noise-free case, it follows that $(2-\gamma)^{F_i} = \prod_{e \in \mathcal{E}_i} w(e)$. When $\alpha_p > 0$, then F_i is the number of times some weight in $\mathcal{E}_i$ is multiplied by either $(2-\gamma)$ or γ. Since the number of times some weight in $\mathcal{E}_i$ is multiplied by γ is J_i we have that

$$(2-\gamma)^{F_i - J_i} \gamma^{J_i} = (2-\gamma)^{F_i} \left(\frac{\gamma}{2-\gamma}\right)^{J_i} = \prod_{e \in \mathcal{E}_i} w(e).$$

Taking logarithms of both sides we obtain the stated result. ■

Note that if $n_i = 1$ then $J_i = |\mathcal{E}_i| = F_i = 0$. The proof of our main theorem uses Lemma 2 as its starting point. We first obtain (Lemma 4) a lower bound for F_i that depends on the total number of mistakes, μ_i, made by our algorithm when making a prediction for an entry in cluster i. Next we must determine the maximum amount by which the "noisy" entries of the submatrix S^i cause the weights in $\mathcal{E}_i$ to be "weakened" (i.e. multiplied by γ) instead of being "strengthened" (i.e. multiplied by $(2 - \gamma)$) as desired. In Lemma 5 we show how J_i can be upper bounded in terms of δ_i, the noise within cluster i. Finally in Lemma 7 we obtain an upper bound for the sum of the logarithms of the weights. We do this by observing that the total weight in the system never increases and then use the convexity of the logarithm function. The proof of our main theorem essentially combines all the lemmas and uses an additional convexity argument for combining the contributions from all clusters.

We now obtain a lower bound for F_i. In order to obtain this lower bound, it is crucial to first obtain an upper bound on the number of mistakes for a given cluster and given force. This quantity characterizes the rate at which the weighted-majority nodes are gaining information.

LEMMA 3 *For each cluster r and force f there are at most m mistakes of force f.*

Proof: We use a proof by contradiction. Suppose that for cluster i the learner makes $m + 1$ force f mistakes. Then there must be two mistakes that occur for the same column. Suppose the first of these mistakes occurs when predicting M_{rj} and the second occurs when predicting $M_{r'j}$ where both rows r and r' are in cluster i. However, after making a force f mistake when predicting M_{rj} that entry is known and thus the force of the $M_{r'j}$ mistake must be at least $f + 1$ giving the desired contradiction. ■

We now compute a lower bound for the force of the mistakes made when predicting entries in cluster i.

LEMMA 4 *For any $1 \le i \le k_p$,*

$$F_i \ge \max\left\{\mu_i - m, \frac{\mu_i^2}{2m} - \frac{\mu_i}{2}\right\}.$$

Proof: We proceed by showing that both expressions above are lower bounds for F_i. Let $\langle x \rangle^m$ denote a sequence containing the symbol x repeated m times. Let σ_i denote the sum of the first μ_i elements of the sequence $\langle 0\rangle^m\langle 1\rangle^m\langle 2\rangle^m\cdots$. From Lemma 3 it follows that $F_i \ge \sigma_i$. Thus, clearly our first lower bound

$$F_i \ge \mu_i - m$$

follows since all but m mistakes have force at least one.

We now compute a more sophisticated lower bound on σ_i. Let $s(x) = \sum_{k=1}^{x} k = \frac{x(x+1)}{2}$. Using the structure illustrated in Figure 5 it is easily seen that

$$F_i \ \ge \ m\, s\left(\left\lceil\frac{\mu_i}{m}\right\rceil - 1\right) - \left(m\left\lceil\frac{\mu_i}{m}\right\rceil - \mu_i\right)\left(\left\lceil\frac{\mu_i}{m}\right\rceil - 1\right)$$

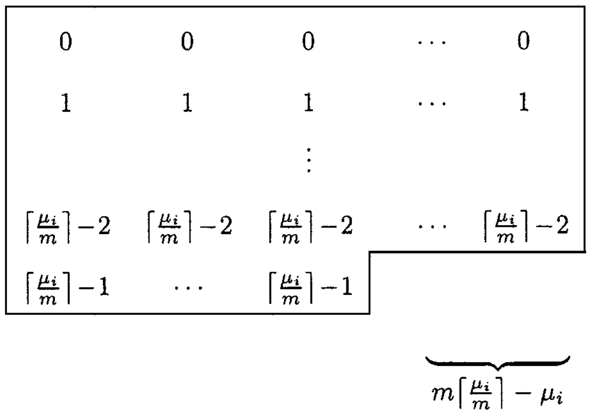
0 0 0 ⋯ 0
1 1 1 ⋯ 1
$\lceil\frac{\mu_i}{m}\rceil-2$ $\lceil\frac{\mu_i}{m}\rceil-2$ $\lceil\frac{\mu_i}{m}\rceil-2$ ⋯ $\lceil\frac{\mu_i}{m}\rceil-2$
$\lceil\frac{\mu_i}{m}\rceil-1$ ⋯ $\lceil\frac{\mu_i}{m}\rceil-1$
$m\lceil\frac{\mu_i}{m}\rceil-\mu_i$

Figure 5. First μ_i elements of the sequence $\langle 0\rangle^m\langle 1\rangle^m\langle 2\rangle^m\ldots$.

$$
\begin{aligned}
&= \left(\left\lceil\frac{\mu_i}{m}\right\rceil - 1\right)\left(\mu_i - \frac{m}{2}\left\lceil\frac{\mu_i}{m}\right\rceil\right)\\
&= \frac{\mu_i^2}{2m} - \frac{\mu_i}{2} + \frac{(1-d)dm}{2}\\
&\geq \frac{\mu_i^2}{2m} - \frac{\mu_i}{2},
\end{aligned}
$$

where $d = \lceil\frac{\mu_i}{m}\rceil - \frac{\mu_i}{m}$ and the last inequality follows from the observation that $0 \leq d < 1$. This completes the proof of the lemma. ■

Observe that the simple linear bound is a better lower bound only for $m < \mu_i < 2m$.

Next, we capture the relationship between J_i and the noise within cluster i of the partition to obtain an upper bound for J_i.

LEMMA 5 *For* $1 \leq i \leq k_p$,

$$J_i \leq \delta_i n_i - \frac{\delta_i^2}{m}.$$

Proof: For ease of exposition, we assume that for each cluster i and column j, the majority of the entries in S_j^i, are 1. Thus $\delta_{i,j}$ is exactly the number of 0's in S_j^i. Observe that for every known 0 entry in S_j^i, the quantity J_i is increased by one whenever the learner makes a prediction error when predicting the value of an entry in S_j^i that is a 1. Thus, in the worst case, each of the $\delta_{i,j}$ entries in S_j^i that are 0 could cause J_i to be incremented for each of the $n_i - \delta_{i,j}$ entries in S_j^i that are 1. Thus,

$$J_i \leq \sum_{j=1}^{m} \delta_{i,j}(n_i - \delta_{i,j}) = \delta_i n_i - \sum_{j=1}^{m} \delta_{i,j}^2.$$

Since x^2 is convex, it follows that $\sum_{j=1}^{m} \delta_{i,j}^2 \geq \frac{\delta_i^2}{m}$. This completes the proof of the lemma. ■

Next we obtain an upper bound on the sum of the logarithms of the weights of a set of edges from $\mathcal{A}$. A key observation used to prove this upper bound is that the overall weight in the system never increases. Therefore, since the initial weight in the system is $n(n-1)/2$ we obtain the following lemma.

LEMMA 6 *Throughout the learning session for any $\mathcal{A}' \subseteq \mathcal{A}$,*

$$\sum_{e \in \mathcal{A}'} w(e) \leq \frac{n(n-1)}{2}.$$

Proof: In trials where no mistake occurs the total weight of all edges $\sum_{e \in \mathcal{A}} w(e)$ clearly does not increase. Assume that a mistake occurs in the current trial. Ignore all weights that are not updated. Of the remaining total weight W that participates in the update let c be the fraction that was placed on the correct bit and $1-c$ be the fraction placed on the incorrect bit. The weight placed on the correct bit is multiplied by $2-\gamma$ and the weight placed on the incorrect bit by γ. Thus the total weight of all edges that participated in the update is $(c(2-\gamma)+(1-c)\gamma)W$ at the end of the trial. Since the latter is increasing in c and $c \leq 1/2$ whenever a mistake occurs, we have that the total of all weights updated in the current trial is at most $(\frac{2-\gamma}{2}+\frac{\gamma}{2})W = W$ at the end of the trial. We conclude that the total weight of all edges also does not increase in trials where a mistakes occurs. Finally since $\mathcal{A}' \subseteq \mathcal{A}$, the result follows. ■

LEMMA 7 *Throughout the learning session for any $\mathcal{A}' \subseteq \mathcal{A}$,*

$$\sum_{e \in \mathcal{A}'} \lg w(e) \leq |\mathcal{A}'| \lg \frac{n(n-1)}{2|\mathcal{A}'|}.$$

Proof: This result immediately follows from Lemma 6 and the concavity of the log function. ■

We are now ready to prove our main result.

THEOREM 2 *For all $\beta \in [0,1)$, Algorithm Learn-Relation when using the parameter $\gamma = \frac{2\beta}{1+\beta}$ makes at most*

$$\min\left\{ k_p m + \min\left\{ \frac{\frac{n^2}{2e}\lg e + \alpha_p(n - \frac{\alpha_p}{km})\lg\frac{1}{\beta}}{\lg\frac{2}{1+\beta}}, \sqrt{\frac{3mn^2\lg k_p + 2\alpha_p(mn-\alpha_p)\lg\frac{1}{\beta}}{\lg\frac{2}{1+\beta}}} \right\}\right\}$$

mistakes in learning a binary-relation where the outside minimum is taken over all partitions p, and k_p denotes the size and α_p the noise of partition p.

Proof: Let p be *any* partition of size k_p and noise α_p. We begin by noting that for any cluster i and column j in partition p, $\delta_{i,j} \leq n_i/2$ and thus $\delta_i = \sum_{j=1}^{m} \delta_{i,j} \leq n_i m/2$. Let $S_1 = \{i \mid n_i = 1\}$, and $S_2 = \{i \mid n_i \geq 2\}$. Clearly the total number of mistakes, μ, can be expressed as

$$\mu = \sum_{i \in S_1} \mu_i + \sum_{i \in S_2} \mu_i.$$

Recall that in Lemma 4 we showed that $F_i \geq \mu_i - m$. Observe that if $n_i = 1$ then $F_i = 0$, and thus it follows that $\sum_{i \in S_1} \mu_i \leq |S_1|m$. As shown below, $\sum_{i \in S_2} \mu_i$ is bounded from above by a function of the form $|S_2|m + x$ where x is some positive quantity. Thus

$$\mu = \sum_{i \in S_1} \mu_i + \sum_{i \in S_2} \mu_i = |S_1|m + |S_2|m + x = k_p m + x.$$

Since it is easily seen that the value of x is maximized when $|S_2| = n$, we will assume throughout the remainder of this section that $n_i \geq 2$ for all i.

As we have discussed, the base of our proof is provided by Lemma 2. We then apply Lemmas 4, 5 and 7 to obtain an upper bound on the number of mistakes made by *Learn-Relation.*

We now proceed independently with the two lower bounds for F_i given in Lemma 4. Applying Lemma 2 with the first lower bound for F_i given in Lemma 4, summing over the clusters in p, and solving for μ yields,

$$\mu = \sum_{i=1}^{k_p} \mu_i \leq k_p m + \frac{b}{a} \sum_{i=1}^{k_p} J_i + \frac{1}{a} \sum_{e \in \mathcal{E}} \lg w(e). \tag{2}$$

From Lemma 5 we know that $J_i \leq \delta_i n_i - \frac{\delta_i^2}{m} \leq \delta_i n - \frac{\delta_i^2}{m}$. It is easily verified that the function $n\delta_i - \frac{\delta_i^2}{m}$ is concave. Thus, combining Jensen's inequality with the fact that $\sum_{i=1}^{k_p} \delta_i = \alpha_p$, we obtain:

$$\sum_{i=1}^{k_p} J_i \leq n\alpha_p - \frac{\alpha_p^2}{k_p m} = \alpha_p \left(n - \frac{\alpha_p}{k_p m} \right). \tag{3}$$

In addition, by applying Lemma 7 with $\mathcal{A}' = \mathcal{E}$ we obtain that:

$$\sum_{e \in \mathcal{E}} \lg w(e) \leq |\mathcal{E}| \lg \frac{n(n-1)}{2|\mathcal{E}|}.$$

Next observe that the function $x \lg \frac{n(n-1)}{2x}$ is concave and obtains its maximum value at $x = \frac{n(n-1)}{2e}$. Thus we obtain that

$$|\mathcal{E}| \lg \frac{n(n-1)}{2|\mathcal{E}|} \leq n(n-1)\frac{\lg e}{2e}. \tag{4}$$

Finally by combining Inequalities (2), (3), and (4) we obtain that:

$$\mu \le k_p m + \frac{b}{a}\alpha_p\left(n - \frac{\alpha_p}{k_p m}\right) + \frac{n(n-1)}{2a}\frac{\lg e}{e} \le k_p m + \frac{b}{a}\alpha_p\left(n - \frac{\alpha_p}{k_p m}\right) + \frac{n^2}{2a}\frac{\lg e}{e}$$

proving our first bound on μ.

We now proceed by combining Lemma 2 with the more sophisticated second lower bound for F_i given in Lemma 4 to obtain:

$$\frac{\mu_i^2}{2m} - \frac{\mu_i}{2} \le \frac{b}{a}J_i + \frac{1}{a}\sum_{e\in\mathcal{E}_i} \lg w(e). \tag{5}$$

Next we apply Lemma 7 with $\mathcal{A}' = \mathcal{E}_i$ to obtain:

$$\sum_{e\in\mathcal{E}_i} \lg w(e) \le |\mathcal{E}_i| \lg \frac{n(n-1)}{2|\mathcal{E}_i|}.$$

Applying this above inequality, the inequality $\sqrt{a+b} \le \sqrt{a} + \sqrt{b}$, and Inequality (5) yields

$$\begin{aligned}
\mu_i &\le \frac{m}{2} + \sqrt{\frac{2bm}{a}J_i + \frac{2m}{a}\sum_{e\in\mathcal{E}_i} \lg w(e) + \frac{m^2}{4}} \\
&\le \frac{m}{2} + \sqrt{\frac{2bm}{a}}\sqrt{J_i + \frac{1}{b}|\mathcal{E}_i| \lg \frac{n(n-1)}{2|\mathcal{E}_i|}} + \sqrt{\frac{m^2}{4}} \\
&\le m + \sqrt{\frac{2bm}{a}}\sqrt{J_i + \frac{1}{b}|\mathcal{E}_i| \lg \frac{n(n-1)}{2|\mathcal{E}_i|}}.
\end{aligned}$$

Next we apply Lemma 5 and the fact that $|\mathcal{E}_i| = n_i(n_i-1)/2 \le n_i^2/2$, and then sum over the k_p clusters in p to obtain:

$$\mu \le k_p m + \sqrt{\frac{2bm}{a}} \sum_{i=1}^{k_p} \sqrt{\delta_i n_i - \frac{\delta_i^2}{m} + \frac{n_i^2}{2b}\lg\frac{n(n-1)}{n_i(n_i-1)}}.$$

As shown in the appendix, the function $f(\delta_i, n_i) = \sqrt{\delta_i n_i - \frac{\delta_i^2}{m} + \frac{n_i^2}{2b}\lg\frac{n(n-1)}{n_i(n_i-1)}}$ is concave for $n_i \ge 2$ and $\delta_i \le n_i m/2$. Since $\sum_{i=1}^{k_p} n_i = n$ and $\sum_{i=1}^{k_p} \delta_i = \alpha_p$, we can thus apply Jensen's inequality to obtain:

$$\begin{aligned}
\mu &\le k_p m + k_p\sqrt{\frac{2bm}{a}}\sqrt{\frac{\alpha_p n}{k_p^2} - \frac{\alpha_p^2}{k_p^2 m} + \frac{n^2}{2k_p^2 b}\lg\frac{k_p^2(n-1)}{n-k_p}} \\
&= k_p m + \sqrt{\frac{2bm}{a}\alpha_p\left(n - \frac{\alpha_p}{m}\right) + \frac{n^2 m}{a}\lg\frac{k_p^2(n-1)}{n-k_p}}
\end{aligned} \tag{6}$$

Observe that $n \geq k_p$, and furthermore if $n = k_p$ then at most nm mistakes can occur and the upper bound of the theorem trivially holds. Thus without limiting the applicability of our result we can assume that $n \geq k_p + 1$ which in turn implies that $\frac{n-1}{n-k_p} \leq k_p$. Thus we can further simplify Inequality (6) to obtain:

$$\begin{aligned}\mu &\leq k_p m + \sqrt{\frac{3}{a} mn^2 \lg k_p + \frac{2bm}{a} \alpha_p (n - \frac{\alpha_p}{m})} \\ &= k_p m + \sqrt{\frac{3mn^2 \lg k_p + 2\alpha_p(mn - \alpha_p) \lg \frac{1}{\beta}}{\lg \frac{2}{1+\beta}}}\end{aligned}$$

thus giving us our second bound on μ.

The above analysis was performed for any partition $p \in P$. Thus taking the minimum over all partitions in P we get the desired result. ■

As when applying the weighted majority algorithm to a noise-free setting, notice that we obtain the best performance by selecting $\gamma = 0$ (respectively $\beta = 0$). Thus we obtain the following corollary for the case of learning pure relations.

COROLLARY 1 *For the case of learning pure relations where $\alpha_p = \alpha = 0$, the algorithm Learn-Relation with $\gamma = 0$ learns a pure k-binary-relation making at most*

$$km + \min\left\{\frac{n^2}{2e} \lg e, n\sqrt{3m \lg k}\right\}$$

mistakes in the worst-case.

We now apply the results of Cesa-Bianchi, Freund, Helmbold, Haussler, Schapire, and Warmuth (1993) to tune β for the more general case in which the relation is not pure.

THEOREM 3 *For any positive integers k and α, Algorithm Learn-Relation when using $\gamma = \frac{2\beta}{1+\beta}$, where $\beta = g(z)$ and $z = \sqrt{\frac{3mn^2 \ln k}{2\alpha(mn-\alpha)}}$, makes at most*[7]

$$k_p m + \sqrt{3mn^2 \lg k + 4\alpha_p mn \left(1 - \frac{\alpha_p}{mn}\right) + mn\sqrt{24\alpha n \left(1 - \frac{\alpha}{mn}\right) \ln k}}$$

mistakes, where the minimum is taken over all partitions p whose size k_p is at most k and whose noise α_p is at most α.

Before proving the theorem, we give a graphical example to help provide a feel for how this function grows, and we look at its application to a special case. In Figure 6 we plot the mistake bound given in Theorem 3 for the special case in which $m = n = 1000$ and $k_p = k = 50$. We look at how the given mistake bound grows as $\alpha_p = \alpha$ ranges from 0 to 10,000 (so up to 10% of the entries are noisy). It is important to remember that the given plot shows the provable upper bound on the number of mistakes made by

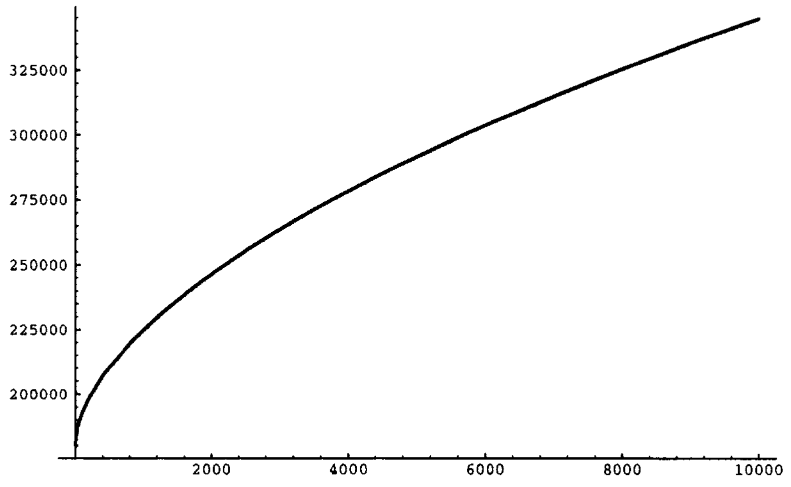

Figure 6. This figure provides a graphical example for the mistake bound of Theorem 3. We have plotted $\alpha = \alpha_p$ on the x-axis, and the value of our upper bound on the mistakes is on the y-axis. In this plot $m = n = 1000$ and $k = k_p = 50$. Observe that there are $1,000,000$ predictions made, and when there is no noise (i.e. $\alpha = 0$) our upper bound on the mistakes is $180,121$.

Learn-Relation (over the $1,000,000$ predictions) — the actual number of mistakes could be lower.

Before proving the theorem, we look at one special case. If partition p is such that $\alpha_p = n$, then the number of mistakes is at most

$$k_p m + \sqrt{3mn^2 \lg k + 4mn^2 + mn^2\sqrt{24 \ln k}}.$$

Proof of Theorem 3: From Theorem 2 we know that for all $\beta \in [0, 1)$, our algorithm makes at most $\min\left\{k_p m + \sqrt{\frac{3mn^2 \lg k_p + 2\alpha_p(mn-\alpha_p)\lg\frac{1}{\beta}}{\lg\frac{2}{1+\beta}}}\right\}$ mistakes where the minimum is taken over all partitions p and k_p denotes the size and α_p the noise of partition p.

Assume that the partition p has the property that $k_p \leq k$ and $\alpha_p \leq \alpha$. Observe that $\left(\ln\frac{1}{\beta}\right) / \left(2\ln\frac{2}{1+\beta}\right) > 1$ for $\beta \in [0, 1)$. Furthermore, since $2\alpha_p(mn - \alpha_p) \leq 2\alpha(mn - \alpha)$ for $\alpha_p \leq \alpha \leq \frac{mn}{2}$, it follows that

$$\frac{3mn^2 \lg k + 2\alpha_p(mn - \alpha_p)\lg\frac{1}{\beta}}{\lg\frac{2}{1+\beta}}$$

$$\begin{aligned}
&\leq \frac{3mn^2 \ln k}{\ln \frac{2}{1+\beta}} + 4\alpha_p(mn - \alpha_p) + 4\alpha(mn - \alpha)\left(\frac{\ln \frac{1}{\beta}}{2\ln \frac{2}{1+\beta}} - 1\right) \\
&= 4\alpha(mn - \alpha)\left(\frac{z^2 + \ln \frac{1}{\beta}}{2\ln \frac{2}{1+\beta}}\right) - 4\alpha(mn - \alpha) + 4\alpha_p(mn - \alpha_p),
\end{aligned}$$

where $z = \sqrt{\frac{3mn^2 \ln k}{2\alpha(mn-\alpha)}}$. So by applying Lemma 1 with and $\beta = g(z)$ we obtain the worst-case mistake bound[8] given in the theorem. ■

In addition to presenting their algorithm to make at most $km + n\sqrt{(k-1)m}$ mistakes, Goldman, Rivest, and Schapire (1989) present an information-theoretic lower bound for a class of algorithms that they call row-filter algorithms. They say that an algorithm A is a *row-filter algorithm* if A makes its prediction for M_{rj} strictly as a function of j and all entries in the set of rows consistent with row r and defined in column j. For this class of algorithms they show a lower bound to $\Omega(n\sqrt{m})$ for $m \geq n$ on the number of mistakes that any algorithm must make. Recently, William Chen (1991) has extended their proof to obtain a lower bound of $\Omega(n\sqrt{m \lg k})$ for $m \geq n \lg k$. Observe that *Learn-Relation* is *not* a row-filter algorithm since the weights stored on the edges between the rows allows it to use the outcome of previous predictions to aid in its prediction for the current trial. Nevertheless, a simple modification of the projective geometry lower bound of Goldman, Rivest, and Schapire (1989) can be used to show an $\Omega(n\sqrt{m})$ lower bound for $m \geq n$ on the number of prediction mistakes by our algorithm. Chen's extension of the projective geometry argument to incorporate k does not extend in such a straightforward manner; however, we conjecture that his lower bound can be generalized to prove that the mistake-bound we obtained for *Learn-Relation* is asymptotically tight. Thus to obtain a better algorithm, more than pairwise information between rows may be needed in making predictions.

6. Concluding Remarks

We have demonstrated that a weighted majority voting algorithm can be used to learn a binary relation even when there is noise present. Our first algorithm uses exponentially many weights. In the noise-free case this algorithm is essentially optimal. We believe that by proving lower bounds for the noisy case (possibly using the techniques developed by Cesa-Bianchi, Freund, Helmbold, Haussler, Schapire, and Warmuth (1993)) one can show that the tuned version of the first algorithm (Theorem 1) is close to optimal in the more general case as well.

The focus of our paper is the analysis of our second algorithm that uses a polynomial number of weights and thus can make predictions in polynomial time. In this algorithm a number of copies of our algorithm divide the problem among themselves and learn the relation cooperatively.

It is surprising that the parallel application of on-line algorithms using multiplicative weight updates can be used to do some non-trivial clustering with provable performance (Theorem 3). Are there other applications where the clustering capability can be exploited? For the problem of learning binary relations the mistake bound of the polynomial algorithm (second algorithm) which uses $\binom{n}{2}$ weights is still far away from the mistake bound of the exponential algorithm (first algorithm) which uses $k^n/k!$ weights. There seems to be a tradeoff between efficiency (number of weights) and the quality of the mistake bound. One of the most fascinating open problem regarding this research is the following: Is it possible to significantly improve our mistake bound (for either learning pure or non-pure relations) by using say $O(n^3)$ weights? Or can one prove, based on some reasonable complexity theoretic or cryptographic assumptions, that no polynomial-time algorithm can perform significantly better than our second algorithm?

Acknowledgments

We thank William Chen and David Helmbold for pointing out flaws in earlier versions of this paper. We also thank the anonymous referees for their comments.

Appendix

We now demonstrate that the function $f(\delta_i, n_i) = \sqrt{\delta_i n_i - \frac{\delta_i^2}{m} + \frac{n_i^2}{2b} \lg \frac{n(n-1)}{n_i(n_i-1)}}$ is concave for $n_i \geq 2$ and $\delta_i \leq n_i m/2$.

For ease of exposition we let $x = \delta_i$ and $y = n_i$. We must now show that $f_{xx} \leq 0$, $f_{yy} \leq 0$, and $f_{xx} f_{yy} - (f_{xy})^2 \geq 0$.

It is easily verified that:

$$f_{xx} = \frac{-y^2}{(f(x,y))^3} \left(\frac{1}{4} + \frac{1}{2mb} \lg \frac{n(n-1)}{y(y-1)} \right) \leq 0.$$

It can also be verified that f_{yy} can be expressed such that the denominator of f_{yy} is

$$16b^2 (\ln 2)^2 (y-1)^2 (f(x,y))^3,$$

and the numerator is

$$\begin{aligned} f_{yy} &= -4y^3(y-1) - y^2 - 4b^2x^2(\ln 2)^2(y-1)^2 - \\ &\quad 4bx \ln 2 \left(y(4y^2 - 7y + 2) - \frac{x}{m}(6y^2 - 10y + 3) \right) - \\ &\quad \ln \frac{n(n-1)}{y(y-1)} \left(2y^2(2y^2 - 4y + 1) + \frac{8b \ln 2}{m} x^2 (y-1)^2 \right). \end{aligned}$$

It is easily shown that $2y^2 - 4y + 1 \geq 0$ for $y \geq 2$. Observe that $y(4y^2 - 7y + 2) - \frac{x}{m}(6y^2 - 10y + 3) \geq 0$ when

$$x \leq \frac{y}{m} \left(\frac{4y^2 - 7y + 2}{6y^2 - 10y + 3} \right).$$

Furthermore, for $y \geq 2$

$$\frac{4y^2 - 7y + 2}{6y^2 - 10y + 3} \geq \frac{4}{7} > \frac{1}{2}$$

and thus it suffices to have $x \leq ym/2$ which is the case.

Finally, it can be verified that $f_{xx}f_{yy} - (f_{xy})^2$ is

$$\frac{y^2\left(4y(y-1) + 1 + bm\ln 2 + 2bmy\ln 2(y-2) + 2\ln\frac{n(n-1)}{y(y-1)}(2y^2 - 4y + 1)\right)}{16b^2(\ln 2)^2(y-1)^2 m(f(x,y))^4} \geq 0$$

for $y \geq 2$.

This completes the proof that $f(x, y)$ is concave over the desired interval.

Notes

1. The adversary, who tries to maximize the learner's mistakes, knows the learner's algorithm and has unlimited computing power.
2. The halving algorithm is described in more detail in Section 2.
3. Throughout this paper we let lg denote the base 2 logarithm and ln the natural logarithm. Euler's constant is denoted by e.
4. The same result holds if one performs the updates after each trial.
5. A version of this second algorithm is described in detail in Figure 3. For the sake of simplicity it is parameterized by an update the factor $\gamma = 2\beta/(1+\beta)$ instead of β. See Section 5 for a discussion of this algorithm called *Learn-Relation*(γ).
6. We can obtain a bound of half of (1) by either letting the algorithm predict probabilistically in $\{0, 1\}$ or deterministically in the interval $[0, 1]$ (Cesa-Bianchi, Freund, Helmbold, Haussler & Schapire 1993; Kivinen & Warmuth, 1994). In the case of probabilistic predictions this quantity is an upper bound for the expected number of mistakes. And in the case of deterministic predictions in $[0, 1]$, this quantity is an upper bound on the loss of the algorithm (where the loss is just the sum over all trials of the absolute difference between the prediction and the correct value).
7. Recall that $g(z) = \frac{1}{1+2z+\frac{z^2}{2\ln 2}}$ and $g(\infty) = 0$.
8. Again, we can obtain improvements of a factor of 2 by either letting the algorithm predict probabilistically in $\{0, 1\}$ or deterministically in the interval $[0, 1]$ (Cesa-Bianchi, Freund, Helmbold, Haussler, Schapire & Warmuth, 1993; Kivinen & Warmuth, 1994).

References

Angluin, D. (1988). Queries and concept learning. *Machine Learning*, 2(4): pp. 319-342.

Barzdin, J. & Freivald, R. (1972). On the prediction of general recursive functions. *Soviet Mathematics Doklady*, 13: pp. 1224–1228.

Cesa-Bianchi, N., Freund, Y., Helmbold, D., Haussler, D., Schapire, R. & Warmuth, M. (1993). How to use expert advice. *Proceedings of the Twenty Fifth Annual ACM Symposium on Theory of Computing*, pp. 382–391.

Chen, W. (1992). Personal communication.

Goldman, S., Rivest, R. & Schapire, R. (1993). Learning binary relations and total orders. *SIAM Journal of Computing*, 22(5): pp. 1006–1034.

Kivinen, J. & and Warmuth, M. (1994) Using Experts for Predicting Continuous Outcomes. *Computational Learning Theory: Eurocolt '93* pp. 109–120, Oxford University Press, Oxford, ISBN 0-19-853492-2.

Littlestone, N. (1988) Learning when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4): pp. 285-318.

Littlestone, N. (1989) *Mistake Bounds and Logarithmic Linear-threshold Learning algorithms*. PhD thesis, U. C. Santa Cruz.

Littlestone, N., Long, P. & Warmuth, M. (1991) On-line learning of linear functions. *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, pp. 465-475. To appear in *Journal of Computational Complexity*.

Littlestone, N. & Warmuth, M. (1994) The weighted majority algorithm. *Information and Computation*, 108(2): pp. 212–261.

Vovk, V. (1990) Aggregating strategies. *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pp. 371-383. Morgan Kaufmann.

Received January 17, 1994
Accepted November 15, 1994
Final Manuscript January 25, 1995