On-Line Learning of Linear Functions

Nicholas Littlestone*
NEC Research Institute
4 Independence Way
Princeton, NJ 08540

Philip M. Long† and Manfred K. Warmuth‡
UC Santa Cruz
Department of Computer Science
Santa Cruz, CA 95064

Abstract

We present an algorithm for the on-line learning of linear functions which is optimal to within a constant factor with respect to bounds on the sum of squared errors for a worst case sequence of trials. The bounds are logarithmic in the number of variables. Furthermore, the algorithm is shown to be optimally robust with respect to noise in the data (again to within a constant factor).

We also discuss an application of our methods to the iterative solution of sparse systems of linear equations.

1 Introduction

Suppose, for budget purposes, each year each member of a panel of economists predicts the next year's GNP and an advisor to the president wishes to combine their predictions to obtain a single prediction. If we measure the loss for each year as the square of the difference between the advisor's prediction and actual GNP, a reasonable goal for the advisor is to minimize the worst case total loss over the years, assuming that some fixed weighted average of the economists is always reasonably close to the actual GNP. In this paper, we present near-optimal strategies for combining opinions in situations like this.

In more abstract terms, we study the on-line learning of linear functions. We assume that learning proceeds in a sequence of trials. At trial number t the learning algorithm (the advisor) is presented with an *instance* $x^{(t)} \in [0,1]^n$ (the estimates of the n economists, where the GNP is measured in units such that it can never possibly be greater than 1) and is required to return a real number $\lambda^{(t)}$ which is interpreted as a *prediction*. After predicting, the algorithm receives a real number $\rho^{(t)}$ from the world, called a *response*, which can be interpreted as the truth. In the simplest case we consider, $\rho^{(t)} = \mu \cdot x^{(t)}$ for each trial, where μ is a hidden coefficient vector in $[0,1]^n$ whose components sum to 1 and $\cdot$ denotes the dot product. The loss of an algorithm over a sequence of m trials is $\sum_{t=1}^{m}(\lambda^{(t)} - \rho^{(t)})^2$.

We present an algorithm whose worst case loss is at most $2(\ln n - H(\mu))$ where $H(\mu) = -\sum_{i=1}^{n} \mu_i \ln \mu_i$ is the entropy of the hidden coefficient vector. Since for all relevant μ, $H(\mu) \geq 0$, the total loss of our algorithm is at most $2 \ln n$. Also, as μ approaches $(1/n, 1/n, ..., 1/n)$, $H(\mu)$ approaches $\ln n$, and our bounds approach 0. We show that for all values of $H(\mu)$, our algorithm is optimal to within a constant factor. Note that our bounds hold for an arbitrarily large number m of trials.

In reality, there may not be any fixed set of weights such that the corresponding weighted average of economists' estimates always equals the actual GNP. In that case, for any finite sequence of trials, the loss of our algorithm is still bounded by $O(\inf\{\ln n - H(\mu) + \sum_{t=1}^{m}(\mu \cdot x^{(t)} - \rho^{(t)})^2\})$, where the infimum is over all choices of $\mu \in [0,1]^n$ whose

*The research reported here was primarily done while this author was at Harvard supported by ONR grant N00014-85-K-0445 and DARPA grant AFOSR-89-0506. Email address: nickl@research.nj.nec.com.

†Supported by ONR grant N00014-86-K-0454. Email address: plong@saturn.ucsc.edu

‡Supported by ONR grant N00014-86-K-0454. Part of this work was done while employed by IIAS-SIS Fujitsu Limited in Numazu, Japan. Email address: manfred@mira.ucsc.edu.

components sum to one.[1] In particular, this implies that the total loss of our algorithm is $O(\log n + N)$, where N is the total loss obtained from the best fixed weight vector. This performance is obtained even though the algorithm is not given any information about future examples and about the error term (the sum in the above expression). As in the case in which all examples are consistent with some hidden function, we can show that our algorithm is optimal to within a constant factor. We can also give algorithms for more general linear functions defined on more general domains by transforming such problems into the basic problem discussed above. These transformations resemble those studied in [Hau88] [KLPV87] [Lit88] [PW90].

In [Myc88], Mycielski gives worst case bounds on the total loss of an on-line learning algorithm for linear functions. However, in place of our squared loss, he uses a "truncated" loss function, in which if $|\lambda^{(t)} - \rho^{(t)}|$ is smaller than some fixed constant ϵ, the loss for trial t is taken to be 0. Thus results from [Myc88] cannot be directly translated to our model, since an algorithm could conceivably have zero total loss on a sequence of examples using a truncated loss function and have infinite loss using the squared loss function by repeatedly having errors slightly smaller than ϵ.

Our algorithms are motivated by those of [Lit88] [Lit89] for learning simple boolean functions, such as clauses with a small number of literals. In that case the predictions and responses are boolean. A mistake occurs when the prediction and response disagree, and the loss is taken to be the total number of mistakes in all trials. Algorithms are given in those papers for learning k-literal clauses whose worst case mistake bounds are at most a constant factor from optimal. We generalize the techniques developed there to the learning of continuous linear functions. Optimal algorithms for a simple continuous case have already been given in [LW89]. In our notation, this is the case when exactly one of the hidden μ_i's is 1 and the rest are 0.[2]

As in [Lit88] [Lit89] [LW89], our algorithm maintains a vector of n weights that is updated each trial after the response is received. Let $v^{(t)}$ represent this weight vector before trial t. Our algorithm always predicts with the current weight vector: i.e., predicts $\lambda^{(t)} = v^{(t)} \cdot x^{(t)}$. Note that in the noise-free case it is easy to always find a coefficient vector v consistent with the previously observed examples, i.e., such that for all j less than t, $v \cdot x^{(j)} = \rho^{(j)}$. However, consistency is neither necessary nor sufficient to obtain the performance we describe. We can show that an algorithm that predicts using an arbitrary consistent linear function can have loss of $\Omega(n)$. Our algorithm does not necessarily maintain consistency with previously observed examples. Instead, it is designed so that it "learns a lot" from a large loss, so that the cumulative loss is only logarithmic in n instead of linear.

To get some intuition about updates of the weights that might achieve the above, let us go back to our initial example of predicting the GNP. An obvious strategy for the advisor would be to predict with the average estimate of the economists. Suppose, however, the advisor notices that some economists are better at predicting the GNP. A good method for the advisor would be to initially weigh all opinions equally, and adjust the weight assigned to each economist based on her performance.

When using a weighted average for prediction, a natural interpretation of the weights is as the relative "credibilities" of the economists. Given this interpretation, a natural reweighting strategy is to reduce the weights of each economist according to some monotone function of how far off her estimate was (e.g., the Weighted Majority algorithm [LW89]), and then normalize so that the weights sum to one. In the discrete case this approach can lead to logarithmic total mistake bounds [Lit88] [Lit89] [LW89]. Furthermore, it was shown in [LW89] that in the continuous case the loss of the advisor is at most $O(\log n)$ plus a constant times the least individual loss of any of the n economists.[3]

However, if one wishes to learn a linear combinations without assuming that any one economist does well individually, then this strategy does not work. Suppose that there were three economists: one who always wildly overestimated the GNP, one who wildly underestimated the GNP, and one who always gave an estimate slightly greater than the correct GNP. Suppose further that the average of the estimates of the two wild economists was always

[1] There is a subtle trade off between the two summands in the infimum. Even if there is a μ such that $\rho^{(t)} = \mu \cdot x^{(t)}$ for all $1 \leq t \leq m$, the infimum sometimes occurs at a $\tilde{\mu}$ with higher entropy for which $\sum_{t=1}^{m} (\tilde{\mu} \cdot x^{(t)} - \rho^{(t)})^2 > 0$.

[2] These results are with respect to the loss function $|\lambda^{(t)} - \rho^{(t)}|$.

[3] Again, these results are with respect to the loss function $|\lambda^{(t)} - \rho^{(t)}|$.

exactly correct, so that there exists a weighting with zero loss. It is easy to see that in this example the loss of the above strategy is unbounded: the wild people's contribution will be steadily decreased and in the limit the prediction of the economist who is always slightly off will dominate.

It turns out that the following intuition can be translated into an essentially optimal learning algorithm. If the aggregate opinion was greater than the true GNP, then those whose predictions were too small were "pulling" the aggregate in the right direction, and the marginal effect of increasing their weights is to improve the aggregate prediction, even if their predictions were very inaccurate. Thus one would want to increase the weights of those whose predictions were too small, and decrease the weights of those whose predictions were too large. Of course, these changes are reversed when the aggregate prediction is too small.

Our algorithm uses the above philosophy of updating the weights with the additional crucial feature that the smaller the aggregate error, the "gentler" the updates. In particular, if the aggregate prediction is correct, the weights are not changed.

As is done in [Lit89] for linear threshold functions, we use the relative entropy between our weights and a target set of weights as a measure of progress. The relative entropy is an information theoretic notion normally used to measure the distance between probability distributions. (Though the formulas for relative entropy and entropy are both important in our work, we do not know of a natural way of interpreting the weights used in our algorithm and those of the hidden function as probabilities. They formally resemble probability distributions in that they form vectors of non-negative numbers that sum to 1.)

Linear functions are widely used. We expect that our algorithm may become a standard submodule for learning more complicated functions or for learning linear combinations of previously learned functions. We are also investigating the case in which the coefficient vector changes gradually over time, corresponding to a case in which some linear combination of the economists is close to the actual GNP for a certain period, and then in later periods other linear combinations do well. The algorithm is to "track" the best linear combination with some additional cost that grows as a function of how much the coefficient vector changes over time. This would generalize the methods of [LW89] with which one could track the best single economist.

A variant of our learning algorithm leads to an algorithm for iteratively producing estimates $x^{(1)}, x^{(2)}, \ldots$ of the solution of the linear system $Ax = b$ such that, if A^{-1} exists, the estimates converge to $A^{-1}b$. Even if A is square and singular or is even non-square, if the system has a solution, then $Ax^{(1)}, Ax^{(2)}, \ldots$ converges to b. The time per iteration is linear in the number of non-zero entries of A and thus our algorithm is well suited for sparse matrices. Existing algorithms for the iterative solution of sparse linear systems are most effective under restrictive assumptions on the form of the coefficient matrix. The algorithm presented here works for all coefficient matrices.

However, our algorithm in general requires more iterations than the best available iterative methods. We can show that the number of iterations required for the relative error (in the L_2 norm) of the hypothesized solution to be smaller than a given ϵ grows at most polynomially with the number of variables, $1/\epsilon$, and the condition number [GL90] of the coefficient matrix. The number of iterations for the best existing algorithms [GL90] [JM88] [PR85] to converge is at most logarithmic in $1/\epsilon$ and the condition number of the coefficient matrix. It is possible that a better bound on the convergence rate of our algorithm can be obtained with a tighter analysis. In any case, a potentially important contribution of our paper is the introduction of nonstandard methods for iteratively solving sparse linear systems.

2 Preliminaries

Let $\mathbf{R}$ represent the real numbers and $\mathbf{N}$ represent the positive integers. Let log represent the base 2 logarithm, and let ln represent the natural logarithm.

For $x = (x_1, \ldots, x_n) \in \mathbf{R}^n$,

$$||x||_1 = \sum_{i=1}^{n} |x_i|$$

$$||x||_2 = \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{x \cdot x}.$$

For an $m \times n$ real matrix A and $k \in \{1, 2\}$, define

$$||A||_k = \sup \left\{ \frac{||Ax||_k}{||x||_k} : x \in \mathbf{R}^n \right\}.$$

For a nonsingular square matrix A, let the condition number of A, denoted by $cond(A)$, be defined as $||A||_2||A^{-1}||_2$. A matrix's condition number measures how close to singular it is, with a large condition number indicating a nearly singular matrix.

Suppose $\mu, v \in [0,1]^n$ are such that $||\mu||_1 = ||v||_1 = 1$. We define the *entropy* of μ to be $\sum_{i=1}^{n} -\mu_i \ln \mu_i$, where $0 \ln 0$ is understood to be 0, and denote this quantity by $H(\mu)$. The *relative entropy* between v and μ, denoted by $I(\mu||v)$, is given by

$$I(\mu||v) = \sum_{i=1}^{n} \mu_i \ln \frac{\mu_i}{v_i}.$$

For any two such μ and v, it is well known that $I(\mu||v) \geq 0$.

Let X be a set, $Y \subseteq \mathbf{R}$. An *example* for (X, Y) is an element of $X \times Y$. If (x, ρ) is an example, we view ρ as the correct response to the instance x. If f is a function from X to Y, we say f is *consistent* with an example (x, y) if $f(x) = y$, and that f is consistent with a sequence S of examples if it is consistent with each example of S.

A *learning algorithm* A for (X, Y) receives a finite sequence of examples and an additional instance $x \in X$ and outputs a prediction in Y. Formally, A is a mapping from $(X \times Y)^* \times X$ to Y. Let $\mathcal{A}(X, Y)$ be the set of learning algorithms for (X, Y).

Fix X and Y and a learning algorithm A. For a finite sequence of examples $S = \langle (x^{(t)}, \rho^{(t)}) \rangle_{1 \leq t \leq m}$ let $\lambda^{(t)}$ be the *prediction* of A on the t-th example, i.e. $\lambda^{(t)} = A(((x^{(1)}, \rho^{(1)}), ..., (x^{(t-1)}, \rho^{(t-1)})), x^{(t)})$. Then the (square) *loss* of A on S, denoted by $L_A(S)$, is defined to be $\sum_{t=1}^{m} (\lambda^{(t)} - \rho^{(t)})^2$. The loss of A on a particular trial t is $(\lambda^{(t)} - \rho^{(t)})^2$. Finally, if $\mathcal{F}$ is a class of functions from X to Y, let $L_A(\mathcal{F}, N)$ be the supremum of $L_A(S)$ over all finite sequences $S = \langle (x^{(t)}, \rho^{(t)}) \rangle_{1 \leq t \leq m}$ of examples such that there exists $f \in \mathcal{F}$ with $\sum_{t=1}^{m} (f(x^{(t)}) - \rho^{(t)})^2 \leq N$.

We will need the following two simple lemmas. The first follows immediately from Jensen's inequality [Bil86]; the second can be easily verified.

Lemma 1 *For all $\beta > 0, x \in [-1, 1]$,*

$$\beta^x \leq \frac{1}{2}\left(\beta + \frac{1}{\beta}\right) + \frac{1}{2}\left(\beta - \frac{1}{\beta}\right) x.$$

Lemma 2 *For all $x, y \in \mathbf{R}$*

$$x(x - y) \geq \frac{1}{2}(x^2 - y^2).$$

We use a "unit cost" model of computation, in which the usual arithmetic operations on real numbers (addition, multiplication and exponentiation) take unit time. For simplicity, we assume that there is no roundoff error.

3 The basic learning algorithm

The basic learning algorithm is designed to perform well on the set of linear functions defined on $[0,1]^n$ whose coefficients are nonnegative and sum to 1. These functions can be viewed as computing weighted averages.

Our algorithms may be stated formally as follows. We maintain a vector of unnormalized weights which is adjusted after each trial and use the normalized weights for prediction. For each t, let $w^{(t)} \in \mathbf{R}^n$ be the algorithm's weights before trial t, and let $v^{(t)} = w^{(t)}/||w^{(t)}||_1$. Then $\lambda^{(t)}$, the algorithm's prediction of trial t, is given by $v^{(t)} \cdot x^{(t)}$.

Our weights are updated as follows. Choose $\gamma \in (0, 1]$. We have a different algorithm for each choice of γ. We let $w_i^{(1)} = 1$ for all i, and update according to the following rule: given the instance $x^{(t)} = (x_1^{(t)}, ..., x_n^{(t)}) \in [0,1]^n$ and the response $\rho^{(t)} \in [0, 1]$,

$$w_i^{(t+1)} = w_i^{(t)} \beta^{u_t(x_i^{(t)})}$$

where

$$\beta = \sqrt{\frac{1 + \gamma|\lambda^{(t)} - \rho^{(t)}|}{1 - \gamma|\lambda^{(t)} - \rho^{(t)}|}}$$

and $u_t : [0,1] \to [-1,1]$ is defined by

$$u_t(y) = \gamma(\rho^{(t)} - y) sign(\lambda^{(t)} - \rho^{(t)}).$$

In the above, $sign : \mathbf{R} \to \{-1, 1\}$ is defined by

$$sign(y) = \begin{cases} 1 & \text{if } y \geq 0 \\ -1 & \text{if } y < 0. \end{cases}$$

For each $\gamma \in (0, 1]$, call the above algorithm A_γ and let $A = A_1$. A simple inductive argument, which is omitted, verifies that the denominator in the update rule is never 0.

As in [Lit89] in the case of linear threshold algorithms, we use the relative entropy between the coefficient vector μ of a target function and the coefficient vector $v^{(t)}$ of the algorithm's hypothesis as

a measure of progress. Our key lemma relates the change in this measure of progress on a particular trial to the loss of the algorithm on that trial and the square of the difference between the response received and the value of the hidden function. Loosely speaking, this lemma says that the algorithm learns a lot when it makes large errors.

Lemma 3 *Choose $\gamma \in (0,1]$, $n \in \mathbf{N}$. Choose $\mu \in [0,1]^n$, $||\mu||_1 = 1$. Let $\langle (x^{(t)}, \rho^{(t)}) \rangle_{t \in \mathbf{N}}$ be a sequence of examples from $[0,1]^n \times [0,1]$. Let $\langle v^{(t)} \rangle_{t \in \mathbf{N}}$ be the sequence of coefficient vectors hypothesized by A_γ and $\langle \lambda^{(t)} \rangle_{t \in \mathbf{N}}$ be the sequence of A_γ's predictions. Choose $t \in \mathbf{N}$. Let $\Delta_t = I(\mu||v^{(t+1)}) - I(\mu||v^{(t)})$. Then, if $\gamma < 1$,*

$$\Delta_t \leq \frac{\gamma^2 |\mu \cdot x^{(t)} - \rho^{(t)}|\,|\lambda^{(t)} - \rho^{(t)}|}{1-\gamma} - \frac{\gamma^2 (\lambda^{(t)} - \rho^{(t)})^2}{2}.$$

If $\mu \cdot x^{(t)} = \rho^{(t)}$ and $\gamma = 1$,

$$\Delta_t \leq -\frac{(\lambda^{(t)} - \rho^{(t)})^2}{2}.$$

Proof: Let $\langle w^{(t)} \rangle_{t \in \mathbf{N}}$ be the sequence of weight vectors maintained by A_γ. It is easy to see that for each t,

$$\Delta_t = \left[\left(\ln \sum_{i=1}^n w_i^{(t+1)} \right) - \left(\ln \sum_{i=1}^n w_i^{(t)} \right) \right] - \left(\sum_{i=1}^n \mu_i (\ln w_i^{(t+1)} - \ln w_i^{(t)}) \right). \quad (1)$$

Fix t. We have

$$\sum_{i=1}^n w_i^{(t+1)} = \sum_{i=1}^n w_i^{(t)} \beta^{u_t(x_i^{(t)})},$$

which, using Lemma 1, implies that

$$\begin{aligned}
&\frac{\sum_{i=1}^n w_i^{(t+1)}}{\sum_{i=1}^n w_i^{(t)}} = \sum_{i=1}^n v_i^{(t)} \beta^{u_t(x_i^{(t)})} \\
&\leq \sum_{i=1}^n v_i^{(t)} \left[\frac{1}{2}\left(\frac{1}{\beta} + \beta\right) + \frac{1}{2}\left(\beta - \frac{1}{\beta}\right) u_t(x_i^{(t)}) \right] \\
&= \sum_{i=1}^n v_i^{(t)} \frac{1}{2} \left[\left(\frac{1}{\beta} + \beta\right) \right. \\
&\quad \left. + \frac{1}{2}\left(\beta - \frac{1}{\beta}\right) \gamma (\rho^{(t)} - x_i^{(t)}) sign(\lambda^{(t)} - \rho^{(t)}) \right] \\
&= \frac{1}{2}\left(\frac{1}{\beta} + \beta\right) + \frac{1}{2}\left(\frac{1}{\beta} - \beta\right) \gamma |\lambda^{(t)} - \rho^{(t)}|.
\end{aligned}$$

Thus if we let $E = \gamma |\lambda^{(t)} - \rho^{(t)}|$ we have

$$\begin{aligned}
&\left(\ln \textstyle\sum_{i=1}^n w_i^{(t+1)}\right) - \left(\ln \textstyle\sum_{i=1}^n w_i^{(t)}\right) \\
&\leq \ln \left[\frac{1}{2}\left(\frac{1}{\beta} + \beta\right) + \frac{1}{2}\left(\frac{1}{\beta} - \beta\right) \gamma |\lambda^{(t)} - \rho^{(t)}| \right] \\
&\leq \ln \left[\frac{1}{2}\left(\frac{1}{\beta} + \beta\right) + \frac{1}{2}\left(\frac{1}{\beta} - \beta\right) E \right] \\
&= \ln \left[\frac{1}{2}\left(\sqrt{\tfrac{1-E}{1+E}} + \sqrt{\tfrac{1+E}{1-E}} \right) \right. \\
&\quad \left. + \frac{1}{2}\left(\sqrt{\tfrac{1-E}{1+E}} - \sqrt{\tfrac{1+E}{1-E}} \right) E \right] \\
&= \ln \tfrac{1}{2} \left[(1+E)\sqrt{\tfrac{1-E}{1+E}} + (1-E)\sqrt{\tfrac{1+E}{1-E}} \right] \\
&= \ln \left[\sqrt{1+E}\sqrt{1-E} \right] \\
&= \ln \sqrt{1 - \gamma^2 (\lambda^{(t)} - \rho^{(t)})^2} \\
&\leq -\tfrac{1}{2} \gamma^2 (\lambda^{(t)} - \rho^{(t)})^2.
\end{aligned} \quad (2)$$

Next, we have

$$\begin{aligned}
&\textstyle\sum_{i=1}^n \mu_i (\ln w_i^{(t+1)} - \ln w_i^{(t)}) \\
&= \textstyle\sum_{i=1}^n \mu_i \gamma (\rho^{(t)} - x_i^{(t)}) sign(\lambda^{(t)} - \rho^{(t)}) \ln \beta \\
&= (\gamma \ln \beta)(\rho^{(t)} - \mu \cdot x^{(t)}) sign(\lambda^{(t)} - \rho^{(t)}) \\
&\geq -(\gamma \ln \beta) |\mu \cdot x^{(t)} - \rho^{(t)}|.
\end{aligned}$$

Combining the above with (1) and (2) yields

$$\Delta_t \leq (\gamma \ln \beta) |\mu \cdot x^{(t)} - \rho^{(t)}| - \frac{1}{2} \gamma^2 (\lambda^{(t)} - \rho^{(t)})^2.$$

The second inequality of the lemma follows immediately from the above. Now, assume $\gamma < 1$. We have

$$\begin{aligned}
\Delta_t &\leq \left(\gamma \ln \sqrt{\tfrac{1+\gamma|\lambda^{(t)} - \rho^{(t)}|}{1-\gamma|\lambda^{(t)} - \rho^{(t)}|}} \right) |\mu \cdot x^{(t)} - \rho^{(t)}| \\
&\quad - \tfrac{1}{2} \gamma^2 (\lambda^{(t)} - \rho^{(t)})^2 \\
&= \left(\frac{\gamma}{2} \ln \tfrac{1+\gamma|\lambda^{(t)} - \rho^{(t)}|}{1-\gamma|\lambda^{(t)} - \rho^{(t)}|} \right) |\mu \cdot x^{(t)} - \rho^{(t)}| \\
&\quad - \tfrac{1}{2} \gamma^2 (\lambda^{(t)} - \rho^{(t)})^2 \\
&= \left(\frac{\gamma}{2} \ln \left(1 + \tfrac{2\gamma|\lambda^{(t)} - \rho^{(t)}|}{1-\gamma|\lambda^{(t)} - \rho^{(t)}|} \right) \right) |\mu \cdot x^{(t)} - \rho^{(t)}| \\
&\quad - \tfrac{1}{2} \gamma^2 (\lambda^{(t)} - \rho^{(t)})^2 \\
&\leq \tfrac{\gamma^2 |\mu \cdot x^{(t)} - \rho^{(t)}||\lambda^{(t)} - \rho^{(t)}|}{1-\gamma} - \tfrac{1}{2} \gamma^2 (\lambda^{(t)} - \rho^{(t)})^2.
\end{aligned}$$

This completes the proof. □

We can apply the previous lemma to obtain the following loss bounds.

Theorem 4 *Choose $n, m \in \mathbf{N}$. Let $S = \langle (x^{(t)}, \rho^{(t)}) \rangle_{1 \leq t \leq m}$ be any sequence of m examples for $([0,1]^n, [0,1])$. Then for each γ,*

$$L_{A_\gamma}(S) \leq 4 \left(\frac{\ln n}{\gamma^2} + \inf_\mu \left(\frac{N(\mu)}{(1-\gamma)^2} - \frac{H(\mu)}{\gamma^2} \right) \right)$$

where the infimum is over all $\mu \in [0,1]^n$ *with* $||\mu||_1 = 1$ *and for each such* μ, $N(\mu) = \sum_{i=1}^{m}(\mu \cdot x^{(t)} - \rho^{(t)})^2$. *In particular,*

$$L_{A_{0.5}}(S) \leq 16(\ln n + \inf_{\mu}(N(\mu) - H(\mu))).$$

Further, for any sequence $S = \langle (x^{(t)}, \rho^{(t)}) \rangle_{1 \leq t \leq m}$ *S of m examples for* $([0,1]^n, [0,1])$ *such that there exists* $\mu \in [0,1]^n, ||\mu||_1 = 1$ *such that for all* $t, 1 \leq t \leq m$, $\mu \cdot x^{(t)} = \rho^{(t)}$,

$$L_A(S) \leq 2(\ln n - H(\mu)).$$

Proof: We will prove only the first two inequalities. The third follows easily from the second bound of Lemma 3.

Choose $\mu \in [0,1]^n, ||\mu||_1 = 1$. Define $e, q \in [0,1]^m$ as follows. For each t, let $e_t = |\lambda^{(t)} - \rho^{(t)}|$ and let $q_t = |\mu \cdot x^{(t)} - \rho^{(t)}|$, where the $\lambda^{(t)}$'s are generated by A_γ.

Since $I(\mu||v^{(1)}) = \ln n - H(\mu)$ and for all t, $I(\mu||v^{(t)}) \geq 0$, the previous lemma implies that

$$\sum_{t=1}^{m} \frac{\gamma^2 e_t q_t}{1-\gamma} - \frac{\gamma^2 e_t^2}{2} = \frac{\gamma^2 e \cdot q}{1-\gamma} - \frac{\gamma^2 e \cdot e}{2} \geq -\ln n + H(\mu)$$

which implies

$$e \cdot e - \frac{2e \cdot q}{1-\gamma} \leq \frac{2}{\gamma^2}(\ln n - H(\mu)).$$

Since $e \cdot q \leq ||e||_2 ||q||_2$, we have

$$||e||_2^2 - \frac{2||e||_2||q||_2}{1-\gamma} \leq \frac{2}{\gamma^2}(\ln n - H(\mu)).$$

Applying Lemma 2, we get

$$||e||_2^2 - \frac{4||q||_2^2}{(1-\gamma)^2} \leq \frac{4}{\gamma^2}(\ln n - H(\mu))$$

which in turn gives

$$e \cdot e \leq \frac{4}{\gamma^2}(\ln n - H(\mu)) + \frac{4q \cdot q}{(1-\gamma)^2},$$

which implies the first inequality of the theorem. Substituting $\gamma = 0.5$ yields the second inequality:

$$e \cdot e \leq 16(\ln n - H(\mu) + q \cdot q).$$

The third inequality can be easily verified using similar, but simpler, techniques. □

Note that if we choose $v^{(1)}$ to be something other than $(1/n, ..., 1/n)$, reflecting some prior bias on which weighted combination of the experts predicts well, the previous proof can be easily modified to yield bounds of

$$\inf_{\mu} 16(I(\mu||v^{(1)}) + N(\mu))$$

in place of the second bound above, and $2I(\mu||v^{(1)})$ in place of the third bound. Thus, our algorithm can take advantage of increasingly accurate prior beliefs. However, the maximum of the second bound over $\mu \in [0,1]^n$ with $||\mu||_1 = 1$ can be easily seen to be minimized by choosing $v^{(1)} = (1/n, ..., 1/n)$, which confirms the intuition that when one knows nothing about the experts, one should begin by simply taking the average of their predictions.

Note also, that if one has a prior idea of $N(\mu)$ ahead of time, one can tune γ to optimize the first bound of the preceding theorem. Furthermore, lower bounds given later in the paper show that tuning γ can only yield an improvement over the choice $\gamma = 0.5$ by a constant factor.

4 Transformations and more general learning problems

In this section, we generalize the notion of prediction preserving reductions between learning problems for $\{0,1\}$-valued functions [Hau88] [Lit88] [KLPV87] [PW90] to real valued functions and apply these reductions to obtain loss bounds for more general linear functions.

We will need the following definition. Let X and Y be sets, and let $\mathcal{F}$ and $\mathcal{G}$ be families of real-valued functions defined on X and Y respectively. Let $\alpha \geq 0$. We say that $\mathcal{F}$ α-reduces to $\mathcal{G}$ if and only if there is a function $\phi : X \to Y$, called an instance transformation, a function $\psi : \mathcal{F} \to \mathcal{G}$, called a target transformation, and $k \in \mathbf{R}$ such that for all $x \in X, f \in \mathcal{F}$,

$$f(x) = \alpha\psi(f)(\phi(x)) + k.$$

We are now ready for the following theorem, which gives loss bounds for a class of functions in terms of those for a class to which the function can be α-reduced. The proof is straightforward and is omitted due to space constraints.

Theorem 5 *Let X and Y be sets, and let $\mathcal{F}$ and $\mathcal{G}$ be families of real-valued functions defined on X and Y respectively. Let A be an algorithm for Y. Choose $\alpha, N \geq 0$. Then if $\mathcal{F}$ α-reduces to $\mathcal{G}$, there exists an algorithm B for X, such that*

$$L_B(\mathcal{F}, N) \leq \alpha^2 L_A(\mathcal{G}, N/\alpha^2).$$

For each $n \in \mathbf{N}, M, \kappa, c > 0$ we will need the following definitions. Let $WA(n, M, \kappa)$ be the set of $f : [0, M]^n \to [0, M]$ such that there exists $\mu \in [0, 1]^n, ||\mu||_1 = 1$ whose entropy is at least κ such that $f(x) = \mu \cdot x$ for all x. Let $LINEAR(n, M, c)$ be the set of linear functions defined on $[0, M]^n$ such that the sum of the absolute values of their coefficients is at most c. Since the entropy is only defined for non-negative coefficients summing to 1, we omit the entropy parameter from LINEAR.

Let $\mathcal{S}_{\mathrm{WA}}(n, M, \kappa, N)$ be the set of all finite sequences $S = \langle (x^{(t)}, \rho^{(t)}) \rangle_{1 \leq t \leq m}$ of examples in $[0, M]^n \times [0, M]$ such that there is some $f \in WA(n, M, \kappa)$ such that

$$\sum_{t=1}^{m} (f(x^{(t)}) - \rho^{(t)})^2 \leq N.$$

Define $\mathcal{S}_{\mathrm{LINEAR}}(n, M, c, N)$ as the analogous set of sequences of examples in $[0, M]^n \times [-cM, cM]$. Let

$$opt_{\mathrm{WA}}(n, M, \kappa, N)$$

be defined to be

$$\inf\{ \sup \{L_A(S) : S \in \mathcal{S}_{\mathrm{WA}}(n, M, \kappa, N)\} : A \in \mathcal{A}([0, M]^n, [0, M])\}$$

and

$$opt_{\mathrm{LINEAR}}(n, M, c, N)$$

to be

$$\inf\{ \sup \{L_A(S) : S \in \mathcal{S}_{\mathrm{LINEAR}}(n, M, c, N)\} : A \in \mathcal{A}([0, M]^n, [-cM, cM])\}.$$

Next, we apply Theorem 5 to get loss bounds for more general linear functions. The proof consists of giving an M-reduction from $WA(n, M, \kappa)$ to $WA(n, 1, \kappa)$ and a $2cM$-reduction from $LINEAR(n, M, c)$ to $WA(2n + 1, 1, 0)$, then applying the bounds of the previous section. The details are omitted due to space constraints.

Theorem 6

$$\begin{aligned} opt_{\mathrm{WA}}(n, M, \kappa, N) &\in O(M^2(\ln n - \kappa) + N) \\ opt_{\mathrm{LINEAR}}(n, M, c, N) &\in O((cM)^2 \ln n + N) \end{aligned}$$

Using similar techniques, we can easily prove similar theorems for classes formed by linear combinations of functions taken from some finite set, e.g. for bounded degree polynomials. Furthermore, our algorithm can be trivially modified to yield an optimal (to within a constant factor) loss bounded algorithm for the learning problem in which the object hidden from the learner is a matrix, the instances are matrices, and the responses are the results of multiplying the instances with the hidden matrix, where the loss is the sum of the squares of the differences between the entries of the predicted matrix and the true matrix.

5 Lower bounds

We begin by proving a lower bound on $opt_{\mathrm{WA}}(n, 1, \kappa, N)$. Our more general lower bounds can be derived from this initial result. For the proof, we will need the following notation. For $u, v \in \mathbf{N}, v \leq \log u + 1$, let $bit(u, v)$ be the vth least significant bit of the binary representation of u (e.g., $bit(6, 1) = 0, bit(6, 2) = 1, bit(6, 3) = 1$).

Theorem 7 *We have*

$$opt_{\mathrm{WA}}(n, 1, \kappa, N) \geq \frac{(\ln n - \kappa)}{4ln2} + N - \frac{1}{2}.$$

Proof: Consider an adversary which adaptively constructs a sequence of examples as follows. Our adversary consists of two stages. In the first stage, the adversary maintains consistency with some function in $WA(n, 1, k) \subseteq WA(n, 1, \kappa)$. In the second stage, the adversary greedily uses up its "noise budget."

Let $l = \lfloor \log n \rfloor, k = \lceil \kappa/(\ln 2) \rceil$. The instances $x^{(1)}, \ldots, x^{(l-k)}$ of the first stage are constructed as follows: $x_i^{(t)} = 1$ if $bit(i, t) = 1$ and $i \leq 2^k$, otherwise $x_i^{(t)} = 0$. The adversary responds with 1 if the algorithm's prediction is no more than $1/2$, otherwise the adversary responds with 0. Thus the loss of the algorithm on each trial of stage one is at least $1/4$.

Define μ as follows: if $i \leq 2^l$ and for each $t \leq l-k$, $bit(i, t) = \rho^{(t)}$, then let $\mu_i = 2^{-k}$, and otherwise, let

$\mu_i = 0$. Since the number of l bit vectors "satisfying" a $(l-k)$-bit mask is 2^k, $||\mu||_1 = 1$. Also, by construction, the linear function induced by μ is consistent with the examples of the first phase. Trivially, $H(\mu) = k \ln 2 \geq \kappa$. Since the first phase consists of $l-k$ trials, the total loss of the first phase is at least

$$\frac{1}{4}(\lfloor \ln n/(\ln 2) \rfloor - \lceil \kappa/(\ln 2) \rceil). \quad (3)$$

In the second stage, which consists of $\lfloor 4N \rfloor + 1$ trials, each instance is $(1/2, 1/2, ..., 1/2)$, and for the first $\lfloor 4N \rfloor$ trials the adversary simply responds with whichever of 0 or 1 is further from the algorithm's prediction. On the last trial, if the algorithm's prediction is no more than $1/2$, the adversary responds with $1/2 + (1/2)\sqrt{4N - \lfloor 4N \rfloor}$, otherwise he responds with $1/2 - (1/2)\sqrt{4N - \lfloor 4N \rfloor}$.

Let $m = l - k + \lfloor 4N \rfloor + 1$ be the total number of trials of the adversary, Since the fact that, $\mu \cdot (1/2, ..., 1/2)$ must equal $1/2$ implies that for each $t, l-k < t < m$,

$$(\mu \cdot x^{(t)} - \rho^{(t)})^2 = 1/4,$$

we have

$$\sum_{t<m} (\rho^{(t)} - \mu \cdot x^{(t)})^2 = \frac{\lfloor 4N \rfloor}{4}.$$

Also,

$$(\rho^{(m)} - \mu \cdot x^{(t)})^2 = \frac{4N - \lfloor 4N \rfloor}{4},$$

so

$$\sum_{t} (\rho^{(t)} - \mu \cdot x^{(t)})^2 = \frac{4N}{4} = N.$$

Also, the loss on each trial t of phase two is at least $(\mu \cdot x^{(t)} - \rho^{(t)})^2$, thus the total loss of stage two is at least N.

Combining this with (3) yields the desired result. □

Note that this argument is strong in the sense that all of the instances of the sequence of examples, as well as the entropy of the hidden coefficient vector and the amount of noise, may be given to the algorithm before the first prediction is made and adversary can then choose the responses of each example so that the loss is maximized.

Note also that in the case that $\kappa = 0$, the adversary uses only functions with just one nonzero coefficient. This, combined with Theorem 4, implies that the inherent complexity of the problem of learning functions which simply output a selected component is the same (at least to within a constant factor) as that of learning the class of all functions computing weighted averages, which is quite surprising. Classes of weighted-average functions whose weights have high entropy (which requires many non-zero weights) are easier to learn. This is in contrast to the case of learning boolean functions, such as boolean linear-threshold functions, where in general (for classes closed under permutation of the attributes) learning gets harder as the number of relevant variables increases [Lit88] [LW89] [Lit89] [BHL].[4] (Some of the upper bounds of [Lit89] depend on a product of two factors, one of which shows the same decreasing dependence on entropy observed here; that decrease is typically dwarfed by an increase in the other factor as the number of relevant variables increases.)

The following is a straightforward extension of the previous theorem. Its proof is omitted due to space constraints.

Corollary 8 *We have*

$$opt_{\mathrm{WA}}(n, M, \kappa, N) \in \Omega(M^2(\ln n - \kappa) + N)$$
$$opt_{\mathrm{LINEAR}}(n, M, c, N) \in \Omega((cM)^2 \ln n + N)$$

By a least squares algorithm, we mean any algorithm which hypothesizes a linear function at each trial that minimizes the sum of the squared errors on previous examples. Next, we show that a least squares algorithm can have total loss which depends linearly on n.

Theorem 9 *For each $n \in \mathbf{N}$, $M, c, N > 0$, there exists a least squares algorithm B and a sequence $S \in \mathcal{S}_{\mathrm{LINEAR}}(n, M, c, N)$ such that*

$$L_B(S) \geq (cM)^2(n-1) + N.$$

Proof: Choose $n \in \mathbf{N}$, $M, c, N > 0$. As before, the adversary strategy is broken into two stages. In the first stage, the adversary maintains consistency with some element of $LINEAR(n, M, c)$, and in the second stage, the adversary greedily expends a "noise budget."

[4] For certain especially simple classes mistake bounds can again drop as the number of relevant variables becomes a significant fraction of all of the variables.

For the first stage, which consists of $n-1$ examples, the tth instance is given by

$$x_i^{(t)} = \begin{cases} \mathrm{M} & \text{if } i = t \\ 0 & \text{otherwise} \end{cases}$$

and the tth response is always 0. Note that if for each t, $v^{(t)} \in \mathbf{R}^n$ is defined by

$$v_i^{(t)} = \begin{cases} \mathrm{c} & \text{if } i = t \\ 0 & \text{otherwise} \end{cases}$$

then for each $t \leq n-1$, $v^{(t)}$ is consistent with the first $t-1$ examples, and thus minimizes the observed loss on these examples. Yet if $\lambda^{(t)}$ is the prediction made using $v^{(t)}$, then for each t, $\lambda^{(t)} = cM$, thus

$$\sum_{t=1}^{n-1} (\lambda^{(t)} - \rho^{(t)})^2 = (cM)^2(n-1). \qquad (4)$$

Note that $v^{(n)}$ is consistent with all the examples of the first stage.

The second stage is virtually identical to the second stage of Theorem 7, replacing $(1/2, ..., 1/2)$ with $(0, ..., 0)$, and responding with whichever of $-cM$ and cM is farthest from the algorithm's prediction. One can easily see that, as in Theorem 7, the algorithm can be forced to have total loss of N in the second stage. Combining this with (4) yields the desired result. □

As discussed in the introduction, this result suggests that our algorithm is to be preferred in practice in situations where little is known about the generation of examples, in particular, the generation of the noise.

6 Iterative solution of linear systems

In this section, we present an iterative algorithm for the approximate solution of systems of linear equations which is based on the learning algorithm treated in the previous sections.

As in the learning case, we have a basic algorithm which solves special types of systems, and we obtain a general algorithm by transforming instances into this special form. The following is an algorithm which, given an $m \times n$ matrix A and a $m \times 1$ vector b such that the L_2 norm of each of the columns of A and of b is no more than $1/2$, produces a sequence $x^{(1)}, x^{(2)}, ...$ of $n \times 1$ column vectors.

Let $A_1, ..., A_n$ be the columns of A. The algorithm computes a sequence $w^{(1)}, w^{(2)}, ...$ of elements of $\mathbf{R}^n$ as follows. Let $w_i^{(1)} = 1$ for all i. For each t and i, let

$$w_i^{(t+1)} = w_i^{(t)} \left(\frac{1 + ||Ax^{(t)} - b||_2}{1 - ||Ax^{(t)} - b||_2} \right)^{\frac{1}{2} u_t(A_i)}$$

where for each $t \in \mathbf{N}$, $u_t : \mathbf{R}^n \to \mathbf{R}$ is defined by

$$u_t(y) = \begin{cases} \frac{(y-b)\cdot(b-Ax^{(t)})}{||b-Ax^{(t)}||_2} & \text{if } Ax^{(t)} \neq b \\ 1 & \text{otherwise.} \end{cases}$$

For each i and t, let

$$x_i^{(t)} = \frac{w_i^{(t)}}{\sum_{i=1}^n w_i^{(t)}}.$$

In the above, the columns $A_1, ..., A_n$ of the matrix correspond to the components $x_1^{(t)}, ..., x_n^{(t)}$ of the instances of the learning algorithm, each $x^{(t)}$ corresponds to a $v^{(t)}$, and b corresponds to a response $\rho^{(t)}$.

Note that

$$u_t(A_i) = \frac{A_i \cdot (b - Ax^{(t)}) - b \cdot (b - Ax^{(t)})}{||b - Ax^{(t)}||_2}$$

so if $b \cdot (b - Ax^{(t)})$ and $||b - Ax^{(t)}||_2$ are precomputed and a list data structure is used for representing each A_i, $u_t(A_i)$ can be computed in time proportional to the number of nonzero entries in A_i. Note further that, if A is represented using an appropriate data structure,[5] $Ax^{(t)}$ can be computed in time depending on the number of nonzero entries in A, so an entire iteration can be computed in time depending on the number of nonzero entries in A.

First, we obtain bounds on the performance of our algorithm on a very restricted class of linear systems. More general results will be obtained by transforming problems into this restricted form. The following lemma is the basis for our analysis of this algorithm.

Lemma 10 *Let A be an $m \times n$ matrix and b be a $m \times 1$ column vector such that there exists $x \in [0,1]^n, ||x||_1 = 1$ with $Ax = b$. Suppose further*

[5] Here we assume without loss of generality that A has no rows consisting entirely of zeroes and at most one such column.

that for each $i, 1 \leq i \leq n, ||A_i||_2 \leq 1/2$. *Then if* $x^{(1)}, x^{(2)}, \ldots$ *are defined as above,*

$$\sum_{t=1}^{\infty} ||Ax^{(t)} - b||_2^2 \leq 2 \ln n.$$

Proof: As in Theorem 4, we use $I(x||x^{(t)})$ as a measure of progress. For each t, let $\lambda^{(t)} = Ax^{(t)}$. Thus $\lambda^{(t)}$ is the vector of right hand sides obtained by "plugging in" our proposed solution to the input linear system.

Choose t. From Lemma 3, we have

$$\begin{aligned} &I(x||x^{(t+1)}) - I(x||x^{(t)}) = \\ &\quad \left[\left(\ln \textstyle\sum_{i=1}^n w_i^{(t+1)}\right) - \left(\ln \textstyle\sum_{i=1}^n w_i^{(t)}\right)\right] \\ &\quad - \left(\textstyle\sum_{i=1}^n x_i (\ln w_i^{(t+1)} - \ln w_i^{(t)})\right). \end{aligned}$$

Let

$$\beta = \sqrt{\frac{1 + ||\lambda^{(t)} - b||_2}{1 - ||\lambda^{(t)} - b||_2}}.$$

Note again that $\beta \geq 1$. Using a similar line of reasoning as in Lemma 3, we have

$$\begin{aligned} &\textstyle\sum_{i=1}^n w_i^{(t+1)} \\ &\leq \left(\textstyle\sum_{i=1}^n w_i^{(t)}\right) \textstyle\sum_{i=1}^n x_i^{(t)} \left[\frac{1}{2}\left(\frac{1}{\beta} + \beta\right)\right. \\ &\quad \left. + \frac{1}{2}\left(\beta - \frac{1}{\beta}\right) u_t(A_i)\right] \\ &= \left(\textstyle\sum_{i=1}^n w_i^{(t)}\right) \textstyle\sum_{i=1}^n x_i^{(t)} \left[\frac{1}{2}\left(\frac{1}{\beta} + \beta\right)\right. \\ &\quad \left. + \frac{1}{2}\left(\beta - \frac{1}{\beta}\right) \frac{(A_i - b)\cdot(b - \lambda^{(t)})}{||b - \lambda^{(t)}||_2}\right] \\ &= \left(\textstyle\sum_{i=1}^n w_i^{(t)}\right) \left[\frac{1}{2}\left(\frac{1}{\beta} + \beta\right)\right. \\ &\quad \left. + \frac{1}{2}\left(\beta - \frac{1}{\beta}\right) \frac{(\lambda^{(t)} - b)\cdot(b - \lambda^{(t)})}{||b - \lambda^{(t)}||_2}\right] \\ &= \left(\textstyle\sum_{i=1}^n w_i^{(t)}\right) \left[\frac{1}{2}\left(\frac{1}{\beta} + \beta\right)\right. \\ &\quad \left. + \frac{1}{2}\left(\frac{1}{\beta} - \beta\right) ||b - Ax^{(t)}||_2\right]. \end{aligned}$$

As before, this implies that

$$\begin{aligned} &\left(\ln \textstyle\sum_{i=1}^n w_i^{(t+1)}\right) - \left(\ln \textstyle\sum_{i=1}^n w_i^{(t)}\right) \\ &\quad \leq \ln\left[\frac{1}{2}\left(\frac{1}{\beta} + \beta\right) + \frac{1}{2}\left(\frac{1}{\beta} - \beta\right) ||Ax^{(t)} - b||_2\right]. \end{aligned}$$

Next, we have

$$\begin{aligned} &\textstyle\sum_{i=1}^n x_i (\ln w_i^{(t+1)} - \ln w_i^{(t)}) \\ &\quad = \textstyle\sum_{i=1}^n x_i u_t(A_i) \ln \beta \\ &\quad = (\ln \beta) \textstyle\sum_{i=1}^n x_i \frac{(A_i - b)\cdot(b - Ax^{(t)})}{||b - \lambda^{(t)}||_2} \\ &\quad = 0 \end{aligned}$$

since $\sum x_i A_i = b$. Hence $I(x||x^{(t+1)}) - I(x||x^{(t)})$ is no more than

$$\ln\left[\frac{1}{2}\left(\frac{1}{\beta} + \beta\right) + \frac{1}{2}\left(\frac{1}{\beta} - \beta\right) ||Ax^{(t)} - b^{(t)}||_2\right].$$

Setting $E = ||Ax^{(t)} - b^{(t)}||_2$ and using the same technique as in Lemma 3, we get

$$I(x||x^{(t+1)}) - I(x||x^{(t)}) \leq -\frac{1}{2}||Ax^{(t)} - b||_2^2.$$

The theorem follows from the fact that $I(x||x^{(1)}) \leq \ln n$ and $I(x||x^{(t)}) \geq 0$ for all t. □

Next, in the general case in which the system has a solution, we bound the rate of convergence of the residuals produced by our algorithm to the zero vector. The transformation required for this result is similar to that of Section 4, but the details are omitted due to space constraints.

Theorem 11 *There is an algorithm which, given an* $m \times n$ *matrix* A *and a* $m \times 1$ *column vector* b *such that there exists* x *with* $Ax = b$ *and given* $\epsilon > 0$, *produces a vector* $\hat{x}$ *such that* $||A\hat{x} - b||_2 \leq \epsilon$ *in*

$$O\left(k(1 + \log ||x||_1)(\log n)\left(\frac{||A||_2||x||_1}{\epsilon}\right)^2\right)$$

time if $||x||_1 \geq 1$ *and*

$$O\left(k(\log n)\left(\frac{||A||_2}{\epsilon}\right)^2\right)$$

time otherwise, where k *is the number of nonzero entries in* A.

The dependence of the time bound on $||A||_2$ and $||x||_1$ is not surprising, since an absolute measure of error is used.

Finally, we can bound the rate of convergence to the solution in the case that A is a nonsingular square matrix.

Corollary 12 *There is an algorithm that, given an* $n \times n$ *nonsingular matrix* A *with* k *nonzero entries and an* $n \times 1$ *column vector* b *produces a sequence of approximations to the solution* x *of* $Ax = b$, *such that it will produce an approximation* $\hat{x}$ *satisfying* $||\hat{x} - A^{-1}b||_2 / ||A^{-1}b||_2 \leq \epsilon$. *in*

$$O\left(\frac{(kn \log n)(1 + \log ||A^{-1}b||_1) cond(A)^2}{\epsilon^2}\right)$$

time if $||A^{-1}b||_2 \geq 1$ *and*

$$O\left(\frac{(kn\log n)cond(A)^2}{||A^{-1}b||_1\epsilon^2}\right)$$

time otherwise.

Our analysis of the time required by our algorithm is presently somewhat crude, so its performance in practice might far outstrip the best bounds we are presently able to prove. Also, our algorithm for linear systems is a straightforward extension of our learning algorithm. Perhaps more refined application of the ideas encompassed here might lead to improved iterative algorithms for the solution of linear systems.

7 Acknowledgments

We'd like to thank Naoki Abe, Yoav Freund, David Haussler, David Helmbold, Victor Pan, Victor Pereyra, John Reif and Richard Snyder for valuable conversations.

References

[BHL] A. Blum, L. Hellerstein, and N. Littlestone. Learning in the presence of finitely many or infinitely many irrelevant attributes. Unpublished manuscript. Feb, 1991.

[Bil86] P. Billingsley. *Probability and Measure.* John Wiley and Sons, 1986.

[GL90] G.H. Golub and C.F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, 1990.

[Hau88] D. Haussler. Learning conjunctive concepts in structural domains. Technical report, UC Santa Cruz, 1988.

[JM88] W.D. Joubert and T.A. Manteuffel. Iterative methods for nonsymmetric linear systems. *Proceedings of the Conference on Iterative Methods for Large Linear Systems*, pages 149–171, 1988.

[KLPV87] M. Kearns, M. Li, L. Pitt, and L.G. Valiant. On the learnability of boolean formulae. *Proceedings of the 19th Annual Symposium on the Theory of Computation*, pages 285–295, 1987.

[Lit88] N. Littlestone. Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[Lit89] N. Littlestone. *Mistake Bounds and Logarithmic Linear-threshold Learning Algorithms.* PhD thesis, UC Santa Cruz, 1989.

[LW89] N. Littlestone and M.K. Warmuth. The weighted majority algorithm. *Proceedings of the 30th Annual Symposium on the Foundations of Computer Science*, 1989.

[Myc88] J. Mycielski. A learning algorithm for linear operators. *Proceedings of the American Mathematical Society*, 103(2):547–550, 1988.

[PR85] V. Pan and J. Reif. Efficient parallel solution of linear systems. *Proceedings of the 18th ACM Symposium on the Theory of Computation*, 1985.

[PW90] L. Pitt and M.K. Warmuth. Prediction preserving reducibility. *Journal of Computer and System Sciences*, 41(3), 1990.