# ON-LINE LEARNING OF LINEAR FUNCTIONS

Nicholas Littlestone, Philip M. Long and Manfred K. Warmuth

**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).

**Key words.** Machine learning; computational learning theory; on-line learning; linear functions; worst-case loss bounds; adaptive filter theory.
**Subject classifications.** 68T05.

## 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* $\vec{x}_t \in [0, 1]^n$ (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 $\lambda_t$. 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, $\rho_t = \vec{\mu} \cdot \vec{x}_t$ for each trial, where $\vec{\mu}$ 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 a family of algorithms $\{A_\delta : \delta > 0\}$. We prove that for each $\delta > 0$, the worst case loss of $A_\delta$ is at most $(1 + 2\delta^2)(\ln n - H(\vec{\mu}))/2$, where $H(\vec{\mu}) = -\sum_{i=1}^n \mu_i \ln \mu_i$ is the entropy of the hidden coefficient vector. Thus, by choosing a small enough $\delta$, we can make the bound arbitrarily close to $(\ln n - H(\vec{\mu}))/2$. Since for all relevant $\vec{\mu}$, $H(\vec{\mu}) \geq 0$, the upper bound on the total loss of $A_\delta$ approaches $(\ln n)/2$ as $\delta$ approaches 0. Also, as $\vec{\mu}$ approaches $(1/n, 1/n, ..., 1/n)$, $H(\vec{\mu})$ approaches $\ln n$, and our bounds approach 0. We show that for all values of $H(\vec{\mu})$ and choices of $\delta$, $A_\delta$ 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 and any $\delta > 0$, the loss of $A_\delta$ is still bounded by $O(\min\{\ln n - H(\vec{\mu}) + \sum_{t=1}^m (\vec{\mu} \cdot \vec{x}_t - \rho_t)^2\})$, where the minimum is over all choices of $\vec{\mu} \in [0,1]^n$ whose components sum to one. (There is a subtle trade off between the two summands in the minimum. Even if there is a $\vec{\mu}$ such that $\rho_t = \vec{\mu} \cdot \vec{x}_t$ for all $1 \leq t \leq m$, the minimum sometimes occurs at a $\vec{\mu}'$ with higher entropy for which $\sum_{t=1}^m (\vec{\mu}' \cdot \vec{x}_t - \rho_t)^2 > 0$.) In particular, this implies that the total loss of $A_\delta$ 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 algorithms are 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 by Haussler (1989), Kearns *et al.* (1987), Littlestone (1988), and Pitt & Warmuth (1990). For example, we can show that if all $\vec{x}_t$'s have $||\vec{x}_t||_\infty = \max_i |x_{t,i}| \leq M$ and there is a $\vec{\mu}$ such that $||\vec{\mu}||_1 = \sum_{i=1}^n |\mu_i| \leq c$, and $\sum_{t=1}^m (\vec{\mu} \cdot \vec{x}_t - \rho_t)^2 \leq N$, then the transformed algorithm has a sum of squared errors that is $O(c^2 M^2 \log n + N)$.

We show a case where the worst-case total loss of the Widrow-Hoff rule (also sometimes called the delta rule) (Widrow & Hoff 1960, Duda & Hart 1973) is $\Omega(c^2 M^2 n + N)$, where, again, $N$ is the total loss of the best fixed weight vector, and our algorithm has a bound of $O(c^2 M^2 \log n + N)$. On the other hand, one can also show that the Widrow-Hoff rule is within a constant factor of optimal for a closely related problem, where, instead of using $||\cdot||_1$ and $||\cdot||_\infty$ to measure the coefficient vector and the instances, respectively, one uses the Euclidian length for both. If $c_0$ and $M_0$ are the corresponding bounds for the Euclidian length, then the bound on the sum of squared errors obtained is $O(c_0^2 M_0^2 + N)$ (Cesa-Bianchi *et al.* 1993). (Mycielski gives worst case bounds on the total loss of the Widrow-Hoff rule. Instead of giving bounds in terms of $\sum_{t=1}^{m}(\vec{\mu}\cdot\vec{x}_t - \rho_t)^2$, he states his bounds in terms of $m \max_t(\vec{\mu}\cdot\vec{x}_t - \rho_t)^2$(Mycielski 1988).)

When is it advantageous to use the algorithm of this paper as opposed to the Widrow-Hoff algorithm? To attempt to gain some insight, let us consider two simple examples. (A similar comparison, in the context of learning linear *threshold* functions, was undertaken by Littlestone (1989).) For now, we will focus on the case in which there is a $\vec{\mu}$ such that each $\rho_t = \vec{\mu}\cdot\vec{x}_t$, and we will refer to the algorithm presented in this paper as the E-rule. First, in the case in which there is a constant $c$ such that the $\vec{x}_t$'s are chosen from $\{c, -c\}^n$, and $\vec{\mu} \in [-c, c]^n$ has at most a constant $k$ nonzero components, we obtain the following upper bounds on the growth of the sum of squared error bounds using the above bounds:

- The Widrow-Hoff algorithm's is at most $O(n)$.
- The E-rule's is at most $O(\log n)$.

However, suppose the $\vec{x}_t$'s are in $[-c, c]^n$ and have at most a constant $k$ nonzero components, and $\vec{\mu} \in \{-c, c\}^n$. In this case, we obtain the following upper bounds by applying the above:

- The Widrow-Hoff algorithm's is (again) at most $O(n)$.
- The E-rule's is at most $O(n^2 \log n)$.

So there are cases where use of each algorithm is advantageous.

One might argue that $\min_{\vec{\mu}} N(\vec{\mu})$ will tend to be much larger than the number $n$ of variables in practice, and that, rather than haggling over the growth of the error bounds with $n$, we should instead concentrate on reducing the constant on $N(\vec{\mu})$ in our bounds. To an extent, we agree. However, the property of the E-rule that its loss bound grows only logarithmically with the number of irrelevant variables appears to make it more attractive to throw in any variable which might possibly be relevant. In particular, one might consider throwing in simple functions of the original variables as new "super-variables" as in Littlestone (1988), Kearns *et al.* (1987), and Haussler (1989) in the case of boolean functions. Such additions have the effect of increasing $n$ and decreasing $N(\vec{\mu})$ at the same time. In the past, the conventional wisdom dictated that one should be careful to include only relevant variables, possibly due to the lack of algorithms whose performance degraded adequately slowly with the number of irrelevant variables. With the E-rule, one gains the freedom to use more "fringe" variables, whose relevance is questionable, but possible.

Our algorithms are motivated by the algorithms of Littlestone (1988, 1989) 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 linear functions defined on $\mathbf{R}^n$. Algorithms for a simple continuous case which are within a constant factor of optimal have already been given in Littlestone & Warmuth (1994). In our notation, this is the case when exactly one of the hidden $\mu_i$'s is 1 and the rest are 0. (These results are with respect to the loss function $|\lambda_t - \rho_t|$.)

As in the algorithms of Littlestone (1988, 1989), Littlestone & Warmuth (1994) and the Widrow-Hoff rule (Widrow & Hoff 1960, Duda & Hart 1973), our algorithms maintain a vector of $n$ weights that is updated each trial after the response is received. Let $\vec{v}_t$ represent this weight vector before trial $t$. Our algorithms always predict with the current weight vector: i.e., they predict $\lambda_t = \vec{v}_t \cdot \vec{x}_t$. 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 $t$, $\vec{v} \cdot \vec{x}_j = \rho_j$. 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 algorithms do not necessarily maintain consistency with previously observed examples. Instead, they are designed so that they "learn 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 loss bounds, 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 in Littlestone & Warmuth 1994), and then normalize so that the weights sum to one. In the discrete case, this approach can lead to logarithmic total mistake bounds (Littlestone 1988, 1989, Littlestone & Warmuth 1994). Furthermore, it was shown in Littlestone & Warmuth (1994) 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. (Again, these results are with respect to the loss function $|\lambda_t - \rho_t|$.)

However, if one wishes to learn a linear combination 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 exactly correct, so that there was a weighting with zero total 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 algorithms use 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 Littlestone (1989) 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.)

## 2. Preliminaries

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

For $\vec{x} = (x_1, ..., x_n) \in \mathbf{R}^n$, define $||x||_1$ as follows:

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

Also, we will find it necessary to discuss sequences $\vec{x}_1, \vec{x}_2, ...$ of vectors. In such cases, we will refer to the $i$th component of $\vec{x}_t \in \mathbf{R}^n$ as $x_{t,i}$.

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

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

For any two such $\vec{\mu}$ and $\vec{v}$, it is well known that $I(\vec{\mu}||\vec{v}) \geq 0$ and that $I(\vec{\mu}||\vec{v}) = 0$ iff $\vec{\mu} = \vec{v}$.

Let $X$ be a set and $Y \subseteq \mathbf{R}$. An *example* for $(X, Y)$ is an element of $X \times Y$. If $(x, \rho)$ is an example, we view $\rho$ as the correct response to the instance $x$. If $f$ is a function from $X$ to $Y$, we say that $f$ 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 $S$.

Each prediction of an on-line learning algorithm (for $(X, Y)$) is determined by the previous examples and the current instance. Associated with an on-line learning algorithm $A$ we define a mapping of the same name from $(X \times Y)^* \times X$ to $Y$. Let $\mathcal{A}(X, Y)$ be the set of such mappings corresponding to 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 *quadratic loss* of $A$ on $S$ is defined as follows:

$$L_A(S) = \sum_{t=1}^{m} (\lambda_t - \rho_t)^2.$$

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 $S = \langle (x_t, \rho_t) \rangle_{1 \le t \le m}$ of examples for which there exists $f \in \mathcal{F}$ with $\sum_{t=1}^{m} (f(x_t) - \rho_t)^2 \le N$.

We will need the following simple lemmas.

LEMMA 2.1. (KULLBACK 1967) *For* $\lambda, \rho \in [0,1]$,

$$I((\rho, 1-\rho) || (\lambda, 1-\lambda)) \ge 2(\lambda - \rho)^2.$$

LEMMA 2.2. (LITTLESTONE 1989) *For all* $\beta > 0, x \in [0,1]$,

$$\beta^x \le 1 + (\beta - 1)x.$$

*The inequality is an equality iff* $x = 0$ *or* $x = 1$.

The following series of lemmas also give approximations for quantities arising in our analysis. The first can be easily verified.

LEMMA 2.3. *For all* $x, y \in \mathbf{R}$,

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

LEMMA 2.4. *For all* $z, \delta$, *and* $x$ *such that* $\delta > 0$, $0 < z \le 1$, *and* $0 \le x \le 1 - z$,

$$\ln \frac{(x+z+\delta)(1-x+\delta)}{(x+\delta)(1-x-z+\delta)} \le \ln \frac{(z+\delta)(1+\delta)}{\delta(1-z+\delta)}.$$

PROOF. Fix $z, \delta > 0$. Define $f : [0, 1-z] \to \mathbf{R}$ by

$$f(x) = \ln \frac{(x+z+\delta)(1-x+\delta)}{(x+\delta)(1-x-z+\delta)}.$$

Note that it is sufficient to prove that $f$ is convex over its domain, since the right hand side of the claimed inequality is $f(0) = f(1-z)$.

Define $g : [0, 1-z] \to \mathbf{R}$ by

$$g(x) = \ln \frac{x+z+\delta}{x+\delta}.$$

Then, the following equality holds:

$$f(x) = g(x) + g((1-z)-x).$$

Hence, the result follows from the convexity of $g$, which is easily verified. □

Lemma 2.5. *For all $\delta > 0$, and $z$ such that $0 \le z \le 1$,*

$$\ln \frac{(z+\delta)(1+\delta)}{\delta((1+\delta)-z)} \le \frac{(2\delta+1)z}{\delta(1+\delta)}.$$

Proof. Fix $\delta > 0$. Define $f : [0,1] \to \mathbf{R}$ by

$$f(z) = \frac{(2\delta+1)z}{\delta(1+\delta)} - \ln \frac{(z+\delta)(1+\delta)}{\delta((1+\delta)-z)}.$$

We have

$$f'(z) = \frac{2\delta+1}{\delta(1+\delta)} - \frac{2\delta+1}{(z+\delta)(1+\delta-z)} \ge 0.$$

Thus, $f$ is monotonically increasing and is minimized when $z = 0$. The fact that $f(0) = 0$ then completes the proof. □

## 3. The basic learning algorithm

The basic learning algorithm $A_\delta$ 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. Intuitively, the larger $\delta$ is, the more robust the algorithm is against noise, and, correspondingly, the more slowly the algorithm learns.

The algorithm $A_\delta$, where $\delta > 0$ is a parameter, may be stated formally as follows. We maintain a vector of normalized weights which is updated at the end of each trial. For each $t$, let $\vec{v}_t \in [0,1]^n$ be the algorithm's weights before trial $t$. When given the instance $\vec{x}_t = (x_{t,1}, ..., x_{t,n}) \in [0,1]^n$ at trial $t$, the algorithm predicts with $\lambda_t = \vec{v}_t \cdot \vec{x}_t$. Let $\rho_t \in [0,1]$ be the response at trial $t$.

We initialize the weight vector to $\vec{v}_{1,i} = 1/n$ for all $i$. At the end of each trial each weight is multiplied by a factor that depends on $\delta$:

$$v_{t+1,i} = \frac{v_{t,i}\mathrm{fact}(\beta_t, x_{t,i})}{\sum_{i=1}^n v_{t,i}\mathrm{fact}(\beta_t, x_{t,i})},$$

where $\mathrm{fact}(\beta_t, x_{t,i}) \in [\beta_t^{\frac{x_{t,i}+\delta}{1+2\delta}}, 1 + (\beta_t - 1)\frac{x_{t,i}+\delta}{1+2\delta}]$ (any value in this range may be chosen by an implementor of the algorithm) and $\beta_t = \left(\frac{\rho_t+\delta}{\lambda_t+\delta}\right)\left(\frac{1-\lambda_t+\delta}{1-\rho_t+\delta}\right)$. Note that since $\frac{x_{t,i}+\delta}{1+2\delta} \in (0,1)$, Lemma 2.2 assures us that the interval in which $\mathrm{fact}(\beta_t, x_{t,i})$ must lie has positive length.

As in Littlestone (1989) in the case of linear threshold algorithms, we use the relative entropy between the coefficient vector $\vec{\mu}$ of a target function and the coefficient vector $\vec{v}_t$ 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. Loosely speaking, it says that the algorithm learns a lot when it makes large errors.

LEMMA 3.1. *Choose $\delta > 0$ and $n \in \mathbf{N}$. Choose $\vec{\mu} \in [0,1]^n$ such that $||\vec{\mu}||_1 = 1$. Let $\langle(\vec{x}_t, \rho_t)\rangle_{t\in\mathbf{N}}$ be a sequence of examples from $[0,1]^n \times [0,1]$. Let $\langle\vec{v}_t\rangle_{t\in\mathbf{N}}$ be the sequence of coefficient vectors hypothesized by $A_\delta$ and $\langle\lambda_t\rangle_{t\in\mathbf{N}}$ be the sequence of $A_\delta$'s predictions. Let $\Delta_t = I(\vec{\mu}||\vec{v}_{t+1}) - I(\vec{\mu}||\vec{v}_t)$ and for $z \in \mathbf{R}$, let $z'$ denote $\frac{z+\delta}{1+2\delta}$. Then, for all $t$, the following inequalities hold:*

$$\begin{aligned}\Delta_t &\le -I((\rho'_t, 1-\rho'_t)||(\lambda'_t, 1-\lambda'_t)) + \frac{\rho_t - \vec{\mu}\cdot\vec{x}_t}{1+2\delta}\ln\beta_t \\ &\le -\frac{2}{(1+2\delta)^2}(\rho-\lambda)^2 + \frac{|\rho_t - \vec{\mu}\cdot\vec{x}_t|\,|\rho_t - \lambda_t|}{\delta(1+\delta)}.\end{aligned}$$

PROOF. Choose $t$. From the definition of $\Delta_t$ and $\mathrm{fact}(\beta_t, x_{t,i})$ and from Lemma 2.2, it follows that

$$\begin{aligned}\Delta_t &= \sum_{i=1}^n \mu_i \ln\frac{v_{t,i}}{v_{t+1,i}} \\ &= \ln(\sum_{i=1}^n v_{t,i}\mathrm{fact}(\beta_t, x_{t,i})) + \sum_{i=1}^n \mu_i \ln\frac{1}{\mathrm{fact}(\beta_t, x_{t,i})} && (3.1) \\ &\le \ln(\sum_{i=1}^n v_{t,i}(1 + (\beta_t - 1)x'_{t,i})) - \sum_{i=1}^n \mu_i x'_{t,i}\ln\beta_t && (3.2)\end{aligned}$$

$$\begin{aligned} &= \ln(1+(\beta_t-1)\lambda'_t) - \frac{\vec{\mu}\cdot\vec{x}_t+\delta}{1+2\delta}\ln\beta_t \\ &= \ln(1+(\beta_t-1)\lambda'_t) - \rho'_t\ln\beta_t + \frac{\rho_t-\vec{\mu}\cdot\vec{x}_t}{1+2\delta}\ln\beta_t. \qquad (3.3)\end{aligned}$$

Note that (3.2) becomes an equality when the components of $\vec{x}_t$ are binary and $\delta$ goes to 0. Since $\beta_t$ can be written as $\frac{\rho'_t}{\lambda'_t}\frac{1-\lambda'_t}{1-\rho'_t}$, we can rewrite expression (3.3) as follows:

$$-I((\rho'_t, 1-\rho'_t)||(\lambda'_t, 1-\lambda'_t)) + \frac{\rho_t-\vec{\mu}\cdot\vec{x}_t}{1+2\delta}\ln\beta_t.$$

This leads to the first inequality of the lemma.

Next, we upper bound the last expression by using Lemma 2.1 and replacing the second term with its absolute value, obtaining:

$$\Delta_t \leq -2(\rho'_t-\lambda'_t)^2 + \frac{|\rho_t-\vec{\mu}\cdot\vec{x}_t|\,|\ln\beta_t|}{1+2\delta}. \qquad (3.4)$$

Now, we wish to bound $|\ln\beta|$. First, let us assume that $\lambda_t \leq \rho_t$. Let $z = \rho_t - \lambda_t$. Then,

$$\ln\beta = \ln\frac{(\lambda+z+\delta)(1-\lambda+\delta)}{(\lambda+\delta)(1-\lambda-z+\delta)}.$$

Applying Lemmas 2.4 and 2.5, we get that

$$\ln\beta \leq \frac{(2\delta+1)z}{\delta(1+\delta)} = \frac{(2\delta+1)(\rho_t-\lambda_t)}{\delta(1+\delta)}.$$

By symmetry, when $\rho_t \leq \lambda_t$, if we let $z = \lambda_t - \rho_t$, we obtain the following inequalities:

$$\begin{aligned} \ln\frac{1}{\beta} \leq \frac{(2\delta+1)z}{\delta(1+\delta)} &= \frac{(2\delta+1)(\lambda_t-\rho_t)}{\delta(1+\delta)}, \\ |\ln\beta| \leq \frac{(2\delta+1)z}{\delta(1+\delta)} &= \frac{(2\delta+1)|\rho_t-\lambda_t|}{\delta(1+\delta)}. \end{aligned}$$

Substituting into (3.4) yields the desired result. □

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

THEOREM 3.2. *Choose $n, m \in \mathbf{N}$. Let $S = \langle(\vec{x}_t, \rho_t)\rangle_{1\leq t\leq m}$ be any sequence of $m$ examples for $([0,1]^n, [0,1])$. Then, for each $\delta > 0$,*

$$L_{A_\delta} \leq \min_{\vec{\mu}}\left((1+2\delta)^2(\ln n - H(\vec{\mu})) + \frac{(1+2\delta)^4}{4\delta^2(1+\delta)^2}N(\vec{\mu})\right),$$

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

$$L_{A_{1/\sqrt{2}}}(S) \le 5.83(\ln n + \min_{\vec{\mu}}(N(\vec{\mu}) - H(\vec{\mu}))).$$

*Further, for any sequence $S = \langle(\vec{x}_t,\rho_t)\rangle_{1\le t\le m}$ of $m$ examples for $([0,1]^n,[0,1])$ for which there exists $\vec{\mu}\in[0,1]^n$ such that $||\vec{\mu}||_1 = 1$ and for all $t, 1\le t\le m$, $\vec{\mu}\cdot\vec{x}_t = \rho_t$, we have the following bound:*

$$L_{A_\delta}(S) \le \frac{(1+2\delta)^2}{2}(\ln n - H(\vec{\mu})).$$

Proof. Since $I(\vec{\mu}||\vec{v}_1) = \ln n - H(\vec{\mu})$ and and $I(\vec{\mu}||v_{m+1}) \ge 0$,

$$\sum_{t=1}^m \Delta_t = I(\vec{\mu}||v_{m+1}) - I(\vec{\mu}||\vec{v}_1) \ge -\ln n + H(\vec{\mu}).$$

Thus, using the last bound on $\Delta_t$ given in the previous lemma, we get:

$$\sum_{t=1}^m -\frac{2}{(1+2\delta)^2}(\lambda_t-\rho_t)^2 + \frac{|\rho_t - \vec{\mu}\cdot\vec{x}_t|\,|\rho_t-\lambda_t|}{\delta(1+\delta)} \ge -\ln n + H(\vec{\mu}).$$

In the case when $\vec{\mu}\cdot\vec{x}_t = \rho_t$, the above inequalities simplify and it is easy to get the loss bound stated at the end of the lemma.

To get the remaining bounds, we rewrite the second inequality as follows:

$$\sum_{t=1}^m -\left(\frac{\sqrt{2}|\lambda_t-\rho_t|}{1+2\delta}\right)^2 + \left(\frac{\sqrt{2}|\rho_t-\lambda_t|}{1+2\delta}\right)\left(\frac{(1+2\delta)|\rho_t-\vec{\mu}\cdot\vec{x}_t|}{\sqrt{2}\delta(1+\delta)}\right) \ge -\ln n + H(\vec{\mu})$$

and apply Lemma 2.3, obtaining

$$\sum_{t=1}^m -\frac{1}{(1+2\delta)^2}(\lambda_t-\rho_t)^2 + \frac{(1+2\delta)^2}{4\delta^2(1+\delta)^2}(\rho_t - \vec{\mu}\cdot\vec{x}_t)^2 \ge -\ln n + H(\vec{\mu}).$$

Solving for $\sum_{t=1}^m(\lambda_t-\rho_t)^2$ yields the first loss bound of the theorem:

$$L_{A_\delta} \le (1+2\delta)^2(\ln n - H(\vec{\mu})) + \frac{(1+2\delta)^4}{4\delta^2(1+\delta)^2}N(\vec{\mu}).$$

For the second bound, observe that when $\delta = 1/\sqrt{2}$,

$$(1+2\delta)^2 = \frac{(1+2\delta)^4}{4\delta^2(1+\delta)^2} \le 5.83.$$

This completes the proof. □

**3.1. Choosing an initial weight vector.** If we choose $\vec{v}_1$ to be something other than $(1/n, ..., 1/n)$, reflecting some prior bias on which weighted combination of the experts predicts well, then the bounds in the previous theorem hold if we replace "$\ln n - H(\vec{\mu})$" by "$I(\vec{\mu}||\vec{v}_1)$". Thus, our algorithm can take advantage of increasingly accurate prior beliefs. However, for fixed $z$, the quantity

$$\max_{\vec{\mu}:H(\vec{\mu})=z} I(\vec{\mu}||\vec{v}_1)$$

is minimized by choosing $\vec{v}_1 = (1/n, ..., 1/n)$. This partially confirms the intuition that when one knows nothing about the experts, one should begin by simply taking the average of their predictions.

**3.2. Trading between fit and entropy.** There is a curious trade off between $N(\vec{\mu})$ and $H(\vec{\mu})$ in the upper bound

$$L_{A_{1/\sqrt{2}}}(S) \leq 5.83(\ln n + \min_{\vec{\mu}}(N(\vec{\mu}) - H(\vec{\mu}))).$$

For example, assume the algorithm receives a single example $((1, 0, \cdots, 0), 1)$. Since we require that $\vec{\mu} \in [0, 1]^n$ and $||\vec{\mu}||_1 = 1$, only $\vec{\mu}_1 = (1, 0, \cdots, 0)$ is consistent. The upper bound for $\vec{\mu} = \vec{\mu}_1$ is $5.83 \ln n$, since $N(\vec{\mu}_1) = H(\vec{\mu}_1) = 0$. However, for $\vec{\mu} = (1/n, 1/n, \cdots, 1/n)$ the bound is 5.83, which is still far away from the actual loss on the single example. Notice that the minimum in the loss bound is not achieved at the consistent vector $\vec{\mu}_1$.

**3.3. Choosing $\beta$.** How did we come up with our choice of $\beta_t = \frac{\rho_t+\delta}{\lambda_t+\delta}\frac{1-\lambda_t+\delta}{1-\rho_t+\delta}$ for the algorithm $A_\delta$? Consider the upper bound for $\Delta_t$ given by the inequality (3.3) for the case when $\rho_t = \vec{\mu}\cdot\vec{x}_t$ (Note that this inequality becomes an equality if the components of $\vec{x}_t$ are binary and $\delta$ goes to 0.):

$$\Delta_t \leq \ln(1 + (\beta_t - 1)\lambda'_t) - \rho'_t \ln \beta_t.$$

Our above choice for $\beta_t$ is obtained by minimizing this upper bound for $\Delta_t$, i.e., we maximize our bounds on the decrease of $I(\vec{\mu}||\vec{v}_t)$ caused by the update in trial $t$.

However, there are better choices for $\beta_t$ for the case when $\rho_t = \vec{\mu}\cdot\vec{x}_t$. Assume $\mathrm{fact}(\beta_t, x_{t,i}) = \beta_t^{x'_{t,i}}$. Then, from (3.1) we get

$$\Delta_t = \ln(\sum_{i=1}^{n} v_{t,i}\beta_t^{x'_{t,i}}) - \rho'_t \ln \beta_t.$$

Note that $\exp(\Delta_t) = \sum_{i=1}^n v_{t,i}\beta_t^{x'_{t,i}-\rho'_t}$, and therefore that

$$\frac{\partial \exp(\Delta_t)}{\partial \beta_t} = 0 \text{ iff } \rho_t = \vec{v}_{t+1} \cdot \vec{x}_t, \text{ and}$$
$$\frac{\partial^2 \exp(\Delta_t)}{\partial^2 \beta_t} \geq 0 \text{ when } \frac{\partial \exp(\Delta_t)}{\partial \beta_t} = 0 \text{ and } \beta_t \geq 0.$$

Thus $\exp(\Delta_t)$, and therefore $\Delta_t$, has exactly one minimum when $\beta_t \in [0, \infty]$. Denote the $\beta_t$ at the minimum as $\beta_{t,opt}$. Now if we updated with $\beta_{t,opt}$ and fed $\vec{x}'_t$ to $A_\delta$ after the update was made, the algorithm would predict $\rho_t$. Thus, with the optimum choice for $\beta_t$, the algorithm is in some sense "corrective."

Since we have determined the choice for $\beta_t$ which gives the best bound when $\rho_t = \vec{\mu} \cdot \vec{x}_t$, why not use it? First, we know no closed form for $\beta_{t,opt}$. We can use a number of heuristics for approximating $\beta_{t,opt}$ such as gradient descent, Newton's method or binary search. Another choice is to iterate the update of $A_\delta$ a number of times with the same instance $\vec{x}_t$.

However, even if the computational cost of approximating $\beta_{t,opt}$ is not a deterrent, there is a second reason for not choosing a $\beta_t$ that is too close to $\beta_{t,opt}$. This is illustrated with the following example. Assume there is a long sequence of examples consistent with $\vec{\mu} = (1/2, 1/2)$ except that the first example $((1,0),1)$ is noisy. In this case, in order to be consistent, we must hypothesize $\vec{v}_2 = (1,0)$, effectively choosing $\beta_1 = \infty$. Now all future updates cannot correct the second component of the weight vector of $\vec{v}_2$, leading to an unbounded loss on future examples consistent with $(1/2, 1/2)$.

So in case of noise, it is advantageous to choose $\beta_t$ not too close to $\beta_{t,opt}$ and instead make a less drastic update.

**3.4. Tuning $\delta$.** If one has a prior idea of $N(\vec{\mu})$, one can tune $\delta$ to optimize the first bound of the preceding theorem. Nonetheless, lower bounds given later in the paper show that tuning $\delta$ can only yield an improvement of a constant factor over the choice $\delta = 1/\sqrt{2}$.

**3.5. Noise tolerance.** Note that the smallest we can make the constant on the "noise term" by increasing $\delta$ (at the expense of the term depending on $n$ and $H(\vec{\mu})$) is 4. However, our analysis is somewhat loose, which leaves open the possibility that our algorithm's loss (or that of a related algorithm) is bounded by $k(\ln n - H(\vec{\mu})) + N(\vec{\mu})$ for some constant $k$.

## 4. Transformations and more general learning problems

In this section, we use transformations to obtain loss bounds for more general classes of linear functions. These transformations generalize the prediction preserving reductions that have been used in a similar manner in the learning of $\{0,1\}$-valued functions (Haussler 1989, Littlestone 1988, Kearns *et al.* 1987, Pitt & Warmuth 1990).

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 $\mathcal{F}$ $\alpha$-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 $\alpha$-reduced.

Theorem 4.1. *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 $\mathcal{F}$ $\alpha$-reduces to $\mathcal{G}$, there exists an algorithm $B$ for $X$ such that*

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

Proof. Define $B$ as follows. Given an instance $x$, $B$ feeds $\phi(x)$ to $A$, and if $A$ predicts $\lambda$, $B$ returns $\alpha\lambda + k$. Then, when $B$ gets $\rho$ as a reinforcement, it feeds $(\rho - k)/\alpha$ to $A$.

Choose $f \in \mathcal{F}$, and let $S = \langle (x_t, \rho_t)\rangle_{t\in\mathbf{N}}$ be a sequence of examples. Let $\langle \lambda_t \rangle_{t\in\mathbf{N}}$ be the sequence of predictions made by $A$ on $\langle (\phi(x_t), (\rho_t - k)/\alpha)\rangle_{t\in\mathbf{N}}$. Let

$$N = \sum_{t=1}^{\infty}(f(x_t) - \rho_t)^2.$$

Then, since

$$\begin{aligned}\sum_{t=1}^{\infty}(\psi(f)(\phi(x_t)) - \frac{\rho_t - k}{\alpha})^2 &= \sum_{t=1}^{\infty}(\frac{f(x_t) - k}{\alpha} - \frac{\rho_t - k}{\alpha})^2 \\ &= 1/\alpha^2 \sum_{t=1}^{\infty}(f(x_t) - \rho_t)^2 \\ &= N/\alpha^2,\end{aligned}$$

we have

$$\sum_{t=1}^{\infty}(\lambda_t - \frac{\rho_t - k}{\alpha})^2 \leq L_A(\mathcal{G}, N/\alpha^2),$$

and

$$\begin{aligned}\sum_{t=1}^{\infty}((\alpha\lambda_t + k) - \rho_t)^2 &= \alpha^2 \sum_{t=1}^{\infty}(\lambda_t - \frac{\rho_t - k}{\alpha})^2 \\ &\leq \alpha^2 L_A(\mathcal{G}, N/\alpha^2).\end{aligned}$$

The theorem follows from the fact that $S$ was chosen arbitrarily. □

For each $n \in \mathbf{N}, M, \kappa, c > 0$ we will need the following definitions. Let $\mathrm{WA}(n, M, \kappa)$ be the set of $f : [0, M]^n \to [0, M]$ such that there exists a $\vec{\mu} \in [0,1]^n, ||\vec{\mu}||_1 = 1$, whose entropy is at least $\kappa$ and for which $f(\vec{x}) = \vec{\mu} \cdot \vec{x}$ for all $\vec{x}$. Let $\mathrm{LINEAR}(n, M, c)$ 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 $S = \langle(\vec{x}_t, \rho_t)\rangle_{1 \leq t \leq m}$ of examples in $[0, M]^n \times [0, M]$ such that there is some $f \in \mathrm{WA}(n, M, \kappa)$ for which the following inequality holds:

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

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 $\mathrm{opt}_{\mathrm{WA}}(n, M, \kappa, N)$ 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 $\mathrm{opt}_{\mathrm{LINEAR}}(n, M, c, N)$ 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 4.1 to get loss bounds for more general linear functions.

THEOREM 4.2.

$$\mathrm{opt}_{\mathrm{WA}}(n, M, \kappa, N) \leq M^2 \mathrm{opt}_{\mathrm{WA}}(n, 1, \kappa, N/M^2) \in O(M^2(\ln n - \kappa) + N),$$

$$\begin{aligned}\mathrm{opt}_{\mathrm{LINEAR}}(n, M, c, N) &\leq (2cM)^2 \mathrm{opt}_{\mathrm{WA}}(2n + 1, 1, 0, N/(2cM)^2) \\ &\in O((cM)^2 \ln n + N).\end{aligned}$$

PROOF. We will prove only the second bound. The first can be proved analogously.

Choose $n$, $M$, and $c$ appropriately. We present a $2cM$-reduction from LINEAR$(n, M, c)$ to WA$(2n+1, 1, 0)$. The theorem then follows immediately from Theorem 4.1 and Theorem 3.2.

Define the instance transformation $\phi : [0, M]^n \to [0, 1]^{2n+1}$ by

$$\phi(\vec{x}) = \left(\frac{x_1 + M}{2M}, \ldots, \frac{x_n + M}{2M}, \frac{-x_1 + M}{2M}, \ldots, \frac{-x_n + M}{2M}, \frac{1}{2}\right)$$

and define $\psi$ : LINEAR$(n, M, c) \to$ WA$(2n+1, 1, 0)$ as follows. If $g \in$ LINEAR$(n, M, c)$ is defined by

$$g(\vec{x}) = \sum_{i=1}^{n} \mu_i x_i,$$

then let $\psi(g) = f$, where $f$ is defined by

$$f(\vec{x}) = \sum_{i=1}^{2n+1} \nu_i x_i,$$

where

$$\nu_i = \begin{cases} \mu_i / c & \text{if } i \le n \text{ and } \mu_i \ge 0, \\ -\mu_{i-n}/c & \text{if } n < i \le 2n \text{ and } \mu_{i-n} < 0, \\ 1 - \frac{1}{c} \sum_i |\mu_i| & \text{if } i = 2n+1, \\ 0 & \text{otherwise.} \end{cases}$$

It is straightforward but tedious to verify that $\phi$ and $\psi$ form a $2cM$-reduction from LINEAR$(n, M, c)$ to WA$(2n+1, 1, 0)$, completing the proof. □

Using similar techniques, we can easily prove similar theorems for classes formed by linear combinations of functions taken from some fixed finite set, e.g., for bounded degree polynomials. (However, it is unclear whether the resulting bounds are optimal.) 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 an $m \times n$ matrix, the instances are $l \times m$ matrices, and the responses are the $l \times n$ matrices obtained by 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. This can be accomplished by running several copies of our algorithm in parallel, one for each pair formed by choosing a row from the instances and a column from the hidden matrix.

## 5. Lower bounds

We begin by proving a lower bound on $\mathrm{opt}_{\mathrm{WA}}(n, 1, \kappa, N)$. 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 $\mathrm{bit}(u, v)$ be the $v$th least significant bit of the binary representation of $u$ (e.g., $\mathrm{bit}(6, 1) = 0, \mathrm{bit}(6, 2) = 1, \mathrm{bit}(6, 3) = 1$).

THEOREM 5.1. *The following inequality holds:*

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

PROOF. Let $l = \lfloor \log n \rfloor, k = \lceil \kappa/(\ln 2) \rceil$. 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 $\mathrm{WA}(n, 1, k) \subseteq \mathrm{WA}(n, 1, \kappa)$. In the second stage, the adversary greedily uses up its "noise budget."

The instances $\vec{x}_1, ..., \vec{x}_{l-k}$ of the first stage are constructed as follows: $x_{t,i} = 1$ if $\mathrm{bit}(i, t) = 1$ and $i \leq 2^l$, otherwise $x_{t,i} = 0$. 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 $\vec{\mu}$ as follows: if $i \leq 2^l$ and for each $t \leq l - k$, $\mathrm{bit}(i, t) = \rho_t$, then let $\mu_i = 2^{-k}$; otherwise, let $\mu_i = 0$. Since the number of $l$ bit vectors "satisfying" a $(l - k)$-bit mask is $2^k$, $||\vec{\mu}||_1 = 1$. Also, by construction, the linear function induced by $\vec{\mu}$ is consistent with the examples of the first phase. Trivially, $H(\vec{\mu}) = k \ln 2 \geq \kappa$. Since the first phase consists of $l - k$ trials, the total loss of the first phase is at least

$$\frac{1}{4}(\lfloor \ln n/(\ln 2) \rfloor - \lceil \kappa/(\ln 2) \rceil). \tag{5.1}$$

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 $\vec{\mu} \cdot (1/2, ..., 1/2)$ must equal 1/2 implies that for each $t, l - k <$

$t < m$, we have the following equalities:

$$\begin{aligned}
(\vec{\mu} \cdot \vec{x}_t - \rho_t)^2 &= 1/4 \\
\sum_{t<m} (\rho_t - \vec{\mu} \cdot \vec{x}_t)^2 &= \frac{\lfloor 4N \rfloor}{4} \\
(\rho_m - \vec{\mu} \cdot \vec{x}_m)^2 &= \frac{4N - \lfloor 4N \rfloor}{4} \\
\sum_t (\rho_t - \vec{\mu} \cdot \vec{x}_t)^2 &= \frac{4N}{4} = N.
\end{aligned}$$

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

Combining this with (5.1) yields the desired result. □

Note that this argument proves a stronger result than that stated in the theorem, since 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 3.2, 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 (Littlestone 1988, 1989, Littlestone & Warmuth 1994, Blum *et al.* 1991). (Some of the upper bounds of Littlestone (1989) 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. Also, 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.)

The following is a straightforward extension of the previous theorem. Its proof is therefore omitted.

Corollary 5.2. *The following inclusions hold:*

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

We now prove a lower bound on the worst case sum of squared errors that holds for a whole family of algorithms. Note that in this lower bound the $\infty$-norm is used for scaling the instances, and the 1-norm for scaling the coefficient vector. The lower bound is linear in the dimension $n$ of the instances.

Theorem 5.3. *Choose $M, c, N > 0, n \in \mathbf{N}$. Let $\vec{w}$ be an arbitrary vector in $\mathbf{R^n}$. Let $A$ be any algorithm whose prediction in the first trial equals the dot product of the first instance and $\vec{w}$, and whose prediction in each subsequent trial equals the dot product of the current instance and a vector formed by summing $\vec{w}$ and a linear combination of the past instances. Then, there is a sequence $S = ((\vec{x}_1, \rho_1), ..., (\vec{x}_m, \rho_m)) \in LINEAR(n, M, c, N)$ such that*

$$L_A(S) \geq (c^2M^2)n' + N,$$

*where $n'$ is the largest integer less than or equal to $n$ for which an $n' \times n'$ Hadamard matrix exists.*

Proof. We only prove the result in the case $N = 0$. The proof can be extended to the case $N > 0$ as in Theorem 5.1.

In this proof, we use Hadamard matrices, which are square matrices whose entries are $\pm 1$ and whose rows are mutually orthogonal. We construct a sequence of $n'$ orthogonal instances $\vec{x}_1, ..., \vec{x}_{n'}$ in $\{-M, 0, M\}^n$ by scaling each of the rows of an $n' \times n'$ Hadamard matrix appropriately and extending it with 0's if $n' < n$. We arrange that the first component of each instance equals $M$ by negating the instance if necessary. Note that since the instances are orthogonal, the prediction of $A$ in each trial $t$ will just equal $\vec{w} \cdot \vec{x}_t$. If $\vec{\mu} = (c, 0, 0, ..., 0)$, then the total loss will be $\sum_{t=1}^{n'} (\vec{w} \cdot \vec{x}_t - cM)^2$. If $\vec{\mu} = (-c, 0, 0, ..., 0)$, then the total loss will be $\sum_{t=1}^{n'} (\vec{w} \cdot \vec{x}_t + cM)^2$. From this, it is easy to see that $\vec{\mu}$ can be chosen so that the total loss is at least $n'c^2M^2$, as desired. □

Since Hadamard matrices are known to exist at least for all powers of 2, $n'$ can always be chosen to be at least $n/2$. It has been conjectured that Hadamard matrices exist for all $n$ divisible by 4 (see discussion in Agaian 1985).

Recall that the total loss of our algorithm was $O((cM)^2 \log n + N)$. We now briefly discuss two algorithms that belong to the family for which the lower bound holds.

The Widrow-Hoff (WH) algorithm predicts using an unnormalized weight vector which is updated after each trial, i.e., the algorithm's prediction on trial $t$ is $\vec{w}_t \cdot \vec{x}_t$. The initial weight vector $\vec{w}_1$ can be chosen arbitrarily; subsequent weight vectors are obtained from the examples according to the following rule:

$$\vec{w}_{t+1} = \vec{w}_t + \eta_t(\rho_t - \lambda_t)\vec{x}_t,$$

where $\eta_t$ is a positive parameter called the *learning rate.* Clearly no matter what learning rates are used WH always has the property that its weight vector differs from the initial weight vector by a linear combination of the past examples, as required for application of Theorem 5.3.

By a least squares algorithm, we mean any algorithm which predicts with the dot product of the current instance and a weight vector, where the weight vector $\vec{w}_t$ used to predict at trial $t$ minimizes the sum of the squared errors on the previous $(t-1)$ examples. Note that the choice for $\vec{w}_t$ might not be unique. We consider the least squares algorithm that breaks ties in favor of shorter weight vectors, as suggested, for example, in Strang (1988). The following lemma applies.

LEMMA 5.4. (STRANG 1988) *The linear least squares algorithm which breaks ties in favor of shorter weight vectors (in the Euclidian norm) has the property that its weight vector is initially zero and in later trials, is a linear combination of the previous examples.*

Applying Theorem 5.3 to these two algorithms, we immediately get the following corollary.

COROLLARY 5.5. *Choose $M, c, N > 0$, and $n \in \mathbf{N}$ and let $n'$ be the largest integer less than or equal to $n$ for which an $n' \times n'$ Hadamard matrix exists. For a WH algorithm using any initial weight vector and any choices of the learning rates $\eta_t$, there is a sequence $S = ((\vec{x}_1, \rho_1), ..., (\vec{x}_m, \rho_m)) \in LINEAR(n, M, c, N)$ such that*

$$L_{WH}(S) \geq (c^2M^2)n' + N.$$

*The same lower bound holds for the linear least squares algorithm that breaks ties in favor of shorter weight vectors (in the Euclidian norm).*

## 6. Conclusion

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.

The fact that our algorithm must know a bound on the sum of the absolute values of the coefficients of the target function might make it appear somewhat unattractive to practitioners. However, this problem may be circumvented by application of the Weighted Majority algorithm (Littlestone & Warmuth 1994) to a pool consisting of algorithms that assume various upper bounds on the size of the hidden coefficient vectors. Nevertheless, to simplify the application of our techniques to real-world problems, it would be useful to have a variant of our algorithm for which we can directly obtain bounds similar to our present ones without knowing anything about the hidden coefficients.

Our lower bounds also might be improved. Is it possible that similar lower bounds hold even when the algorithm has more information about the hidden coefficients, or even about the upcoming sequence of examples?

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 Littlestone & Warmuth (1994) with which one could track the best single economist.

In addition, it would be interesting to find algorithms which are optimal with respect to other natural loss functions, in particular, $|\lambda_t - \rho_t|$. Recently, Bernstein (1992) has proved an upper bound for this problem in the absence of noise of $\sqrt{n \log n}$, and a lower bound whose dependence on $n$ is $\sqrt{n}$.

Finally, since our algorithms have a similar flavor to the linear threshold algorithms of Littlestone (1988, 1989), and Littlestone & Warmuth (1994), one might ask whether a similar algorithm is optimal for learning the class containing all linear functions composed with the standard sigmoid function $(1/(1+e^{-x}))$. One can trivially obtain bounds from our results, but they appear to be suboptimal.

## Acknowledgements

We are especially grateful to Nicolò Cesa-Bianchi for helpful discussions relating to Theorem 5.3. We'd also like to thank Naoki Abe, Yoav Freund, David Haussler, David Helmbold, and Richard Snyder for valuable conversations.

The research reported here was primarily done while Nick Littlestone was at Harvard supported by ONR grant N00014-85-K-0445 and DARPA grant AFOSR-89-0506. Philip Long was at UC Santa Cruz supported by ONR grant

N00014-91-J-1162. Manfred Warmuth was supported by ONR grant N00014-91-J-1162; part of this work was done while he was employed by IIAS-SIS Fujitsu Limited in Numazu, Japan.

## References

S. S. Agaian, *Hadamard Matrices and Their Applications.* Number 1168 in Lecture Notes in Mathematics. Springer-Verlag, 1985.

E.J. Bernstein, Absolute error bounds for learning linear functions on line. *Proceedings of the 1992 Workshop on Computational Learning Theory*, 1992, 160–163.

A. Blum, L. Hellerstein, and N. Littlestone, Learning in the presence of finitely many or infinitely many irrelevant attributes. *The 1991 Workshop on Computational Learning Theory*, 1991, 157–166.

N. Cesa-Bianchi, P.M. Long, and M.K. Warmuth, Worst-case quadratic loss bounds for a generalization of the Widrow-Hoff rule. *The 1993 Workshop on Computational Learning Theory*, 1993, 429–438.

R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* Wiley, 1973.

D. Haussler, Learning conjunctive concepts in structural domains. *Machine Learning* **4**(1) (1989), 7–40.

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*, 1987, 285–295.

S. Kullback, A lower bound for discrimination in terms of variation. *IEEE transactions on Information Theory* **13** (1967), 126–127.

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

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

N. Littlestone and M. Warmuth, The weighted majority algorithm. *Information and Computation* (1994). To appear.

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

L. Pitt and M.K. Warmuth, Prediction preserving reducibility. *Journal of Computer and System Sciences* **41**(3) (1990), 430–467.

G. Strang, *Linear Algebra and its Applications.* Harcourt, Brace, Jovanovich, 1988.

B. Widrow and M.E. Hoff, Adaptive switching circuits. *1960 IRE WESCON Convention Record* (1960), 96–104.

Manuscript received May 30, 1992

Nicholas Littlestone
NEC Research Institute
4 Independence Way
Princeton, NJ 08540
`nickl@research.nj.nec.com`

Philip M. Long
Computer Science Department
Duke University, P.O. Box 90129
Durham, NC 27708
`plong@cs.duke.edu`

Manfred K. Warmuth
CIS Board, UC Santa Cruz
Santa Cruz, CA 95064
`manfred@mira.ucsc.edu`