we have

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

Hence,

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

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

A.2 Proof of Lemma 5

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

$$\begin{aligned}f(x) &= g(x) + g((1 - z) - x) \\ f'(x) &= g'(x) - g'((1 - z) - x) \\ f''(x) &= g''(x) + g''((1 - z) - x).\end{aligned}$$

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

A.3 Proof of Lemma 6

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

$$\begin{aligned}f'(z) &= \frac{2\delta + 1}{\delta(1 + \delta)} - \left(\frac{\delta((1 + \delta) - z)}{(z + \delta)(1 + \delta)}\right)\left(\frac{\delta((1 + \delta) - z)(1 + \delta) + (z + \delta)(1 + \delta)\delta}{\delta^2(1 + \delta - z)^2}\right) \\ &= \frac{2\delta + 1}{\delta(1 + \delta)} - \frac{2\delta + 1}{(z + \delta)(1 + \delta - z)} \\ &\geq 0.\end{aligned}$$

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

[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.

[Kul67] S. Kullback. A lower bound for discrimination in terms of variation. *IEEE transactions on Information Theory*, 13:126–127, 1967.

[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.

[WH60] B. Widrow and M.E. Hoff. Adaptive switching circuits. *1960 IRE WESCON Conv. Record*, pages 96–104, 1960.

A Some proofs

A.1 Proof of Theorem 9

Define B as follows. Given an instance x, B feeds $\phi(x)$ to A, and if A predicts λ, B returns $\alpha\lambda + k$. Then, when B gets ρ 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 example-reinforcement pairs. 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

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

7 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 coefficients of the target function might make it appear somewhat unattractive to practitioners. However, this problem may be circumvented using standard doubling tricks while still retaining bounds within a constant factor of those of this paper. Nevertheless, to simplify application of our techniques to real-world problems, it would be useful to have a variant of our algorithm which avoided doubling techniques, and obtained similar bounds without knowing anything about the hidden coefficients.

We also are interested in improving our lower bounds. 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 [LW89] with which one could track the best single economist.

In addition, we would like to find algorithms which are optimal with respect to other natural loss functions, in particular, $|\lambda_t - \rho_t|$.

Finally, since our algorithms have a similar flavor to the linear threshold algorithms of [Lit88] [LW89] [Lit89], 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 are suboptimal.

8 Acknowledgments

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

References

[BHL91] 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.

[CW91] N. Cesa-Bianchi and M.K. Warmuth. A comparison of on-line algorithms for learning linear functions. Manuscript, 1991.

[DH73] R.O. Duda and P.E. Hart. *Pattern Recognition and Scene Analysis.* John Wiley and Sons, 1973.

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

Corollary 18: *There is an algorithm that, given an $n \times n$ nonsingular matrix A with k nonzero entries and an $n \times 1$ column vector $\vec{b}$ produces a sequence of approximations to the solution $\vec{x}$ of $A\vec{x} = \vec{b}$, for which the following statements hold: If $||A^{-1}b|| \geq 1$, in time*

$$O(\frac{(\log n)k\,cond(A)^2}{\epsilon^2})$$

it will produce an approximation $\hat{x}$ satisfying

$$\frac{||\hat{x} - A^{-1}\vec{b}||_2}{||A^{-1}\vec{b}||_1} \leq \epsilon.$$

In time

$$O(\frac{n(\log n)k\,cond(A)^2}{\epsilon^2})$$

it will produce an approximation $\hat{x}$ satisfying

$$\frac{||\hat{x} - A^{-1}\vec{b}||_2}{||A^{-1}\vec{b}||_2} \leq \epsilon$$

Proof: To prove the first statement, note that from the previous corollary, the algorithm will produce an $\hat{x}$ satisfying $||A\hat{x} - \vec{b}||_2 \leq \frac{||A^{-1}\vec{b}||_1\epsilon}{||A^{-1}||_2}$ in time

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

For such an $\hat{x}$ we have

$$\frac{||\hat{x} - A^{-1}\vec{b}||_2}{||A^{-1}\vec{b}||_1} = \frac{||A^{-1}(A\hat{x} - \vec{b})||_2}{||A\hat{x} - \vec{b}||_2}\frac{||A\hat{x} - \vec{b}||_2}{||A^{-1}\vec{b}||_1}$$

$$\leq \frac{||A^{-1}||_2||A\hat{x} - \vec{b}||_2}{||A^{-1}\vec{b}||_1} \leq \epsilon$$

This gives the first result. The first result implies that we can produce an $\hat{x}$ satisfying $\frac{||\hat{x}-A^{-1}\vec{b}||_2}{||A^{-1}\vec{b}||_1} \leq \frac{\epsilon}{\sqrt{n}}$ in time $O(\frac{n(\log n)k\mathrm{cond}(A)^2}{\epsilon^2})$. For such an $\hat{x}$ we have

$$\frac{||\hat{x} - A^{-1}\vec{b}||_2}{||A^{-1}\vec{b}||_2} \leq \frac{||A^{-1}\vec{b}||_1}{||A^{-1}\vec{b}||_2}\frac{\epsilon}{\sqrt{n}} \leq \epsilon$$

This completes the proof.

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.

Corollary 16: *Choose $c > 0$. There is an algorithm which, given an $m \times n$ matrix A and an $m \times 1$ column vector $\vec{b}$ such that there exists $\vec{x}$ with $A\vec{x} = \vec{b}$ such that $||\vec{x}||_1 \leq c$ and given $\epsilon > 0$, produces a vector $\vec{x}$ such that*

$$||A\vec{x} - \vec{b}||_2 \leq \epsilon$$

in

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

time, where k is the number of nonzero entries in A, and $K = \max\{||A_i||_2 : 1 \leq i \leq n\}$.

Proof: We have,

$$(A/4K)(\vec{x}/c) = (\vec{b}/4Kc)$$

and $||\vec{x}/c||_1 \leq 1$. By the previous corollary, we can find $\vec{x}$ such that

$$||(A/4K)\vec{x} - (\vec{b}/4Kc)||_2 \leq \frac{\epsilon}{4Kc}$$

in

$$O(\frac{k \log n}{(\epsilon/4Kc)^2}) = O((k \log n)\left(\frac{Kc}{\epsilon}\right)^2)$$

time. The corollary follows from the fact that

$$||(A(c\vec{x}) - \vec{b})||_2 = Kc||(A/4K)\vec{x} - (\vec{b}/4Kc)||_2. \quad \square$$

Finally, since we have bounds on the number of iterations required for the relative error of the right hand sides to be no greater than ϵ for each assumption of an upper bound on $||\vec{x}||_1$, we don't need to know any bound a priori. An algorithm can guess an upper bound on $||\vec{x}||_1$, and compute the corresponding bound on the number of iterations. If this number of iterations has passed without success, it can double its guess and try again. After at most $\log ||\vec{x}||_1$ such stages, the algorithm will terminate. Since the time bound for the last stage dominates the that for all the previous stages, the following corollary easily follows.

Corollary 17: *There is an algorithm which, given an $m \times n$ matrix A and an $m \times 1$ column vector $\vec{b}$ such that there exists $\vec{x}$ with $A\vec{x} = \vec{b}$ and given $\epsilon > 0$, produces a vector $\vec{x}$ such that*

$$||A\vec{x} - \vec{b}||_2 \leq \epsilon$$

in

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

time if $||\vec{x}||_1 \geq 1$ and

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

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

Proof: Follows immediately from the above comment, together from the fact that $||A||_2 \geq \max\{||A_i||_2 : 1 \leq i \leq n\}$. $\square$

The dependence of the time bound on $||A||_2$ and $||\vec{x}||_1$ 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.

The theorem follows from the fact that $I(\vec{x}||\vec{x}_1) \leq \ln n$ and $I(\vec{x}||\vec{x}_t) \geq 0$ for all t. □

The following sequence of corollaries bounds the rate of convergence of residuals to the zero vector. The first corollary establishes the existence of an efficient algorithm which uses the fact that A and $\vec{b}$ are of a special form. Algorithms for more general systems will be obtained through transformations to this systems of this form.

Corollary 15: *There is an algorithm which, given an $m \times n$ matrix A, an $m \times 1$ column vector $\vec{b}$ and $\epsilon > 0$ such that there exists $\vec{x}$ with $A\vec{x} = \vec{b}$ and $||\vec{x}||_1 \leq 1$, and such that for each $i, 1 \leq i \leq n, ||A_i||_2 < 1/2$, produces a vector $\hat{x}$ such that*

$$||A\hat{x} - \vec{b}||_2 \leq \epsilon$$

in

$$O(\frac{k \ln n}{\epsilon^2})$$

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

Proof: First of all, we transform A by adding a column consisting entirely of zeros and for each column in A, adding a column which is its negation. Clearly, if A' is the result of such a transformation, there exists $\vec{x}'$ with $||\vec{x}'||_1 = 1$ and each $\vec{x}'$ nonnegative such that $A'\vec{x}' = \vec{b}$. Also, it is easily seen how to transform an estimate $\hat{x}'$ of $\vec{x}'$ to an estimate $\hat{x}$ of $\vec{x}$ such that $A'\hat{x}' = A\hat{x}$.

The algorithm is to generate $\vec{x}_1, \vec{x}_2, \ldots$ from A' and $\vec{b}$ as described at the beginning of this section, and after generating each $\vec{x}_t$, to test whether

$$||A'\vec{x}_t - \vec{b}||_2 \leq \epsilon$$

and terminate when this is true. As discussed above, each iteration requires $O(k)$ time, since there are exactly twice as many nonzero elements in A' as in A.

We claim that there are at most

$$\frac{2\log(2n+1)}{\epsilon^2}$$

iterations. Assume the contrary for contradiction. Then for each $t \leq \frac{2\log(2n+1)}{\epsilon^2}$,

$$||A'\vec{x}_t - \vec{b}||_2 > \epsilon$$

which implies

$$\sum_{t=1}^{p} ||A'\vec{x}_t - \vec{b}||_2^2 > 2\log(2n+1)$$

which is a contradiction. This completes the proof. □

In our next corollary, we establish the existence of an algorithm for each a priori assumption of an upper bound on $||\vec{x}||_1$. Our algorithm works by transforming the instance of a problem into an instance satisfying the requirements of the previous corollary, applying the algorithm of the previous corollary, then transforming the solution received.

By applying Lemma 4, we obtain

$$\begin{aligned}
\sum_{i=1}^{n} w_{(t+1),i} &= \sum_{i=1}^{n} w_{t,i}\beta_t^{u_t(A_i)} \\
&= \left(\sum_{i=1}^{n} w_{t,i}\right)\sum_{i=1}^{n} x_{t,i}\beta_t^{u_t(A_i)} \\
&\leq \left(\sum_{i=1}^{n} w_{t,i}\right)\sum_{i=1}^{n} x_{t,i}\left[\frac{1}{2}\left(\frac{1}{\beta_t}+\beta_t\right)+\frac{1}{2}\left(\beta_t-\frac{1}{\beta_t}\right)u_t(A_i)\right] \\
&= \left(\sum_{i=1}^{n} w_{t,i}\right)\sum_{i=1}^{n} x_{t,i}\left[\frac{1}{2}\left(\frac{1}{\beta_t}+\beta_t\right)+\frac{1}{2}\left(\beta_t-\frac{1}{\beta_t}\right)\frac{(A_i-\vec{b})\cdot(\vec{b}-\vec{\lambda}_t)}{||\vec{b}-\vec{\lambda}_t||_2}\right] \\
&= \left(\sum_{i=1}^{n} w_{t,i}\right)\left[\frac{1}{2}\left(\frac{1}{\beta_t}+\beta_t\right)+\frac{1}{2}\left(\beta_t-\frac{1}{\beta_t}\right)\frac{(\vec{\lambda}_t-\vec{b})\cdot(\vec{b}-\vec{\lambda}_t)}{||\vec{b}-\vec{\lambda}_t||_2}\right] \\
&= \left(\sum_{i=1}^{n} w_{t,i}\right)\left[\frac{1}{2}\left(\frac{1}{\beta_t}+\beta_t\right)+\frac{1}{2}\left(\frac{1}{\beta_t}-\beta_t\right)||\vec{b}-A\vec{x}_t||_2\right].
\end{aligned}$$

This implies that

$$\left(\ln\sum_{i=1}^{n} w_{(t+1),i}\right)-\left(\ln\sum_{i=1}^{n} w_{t,i}\right)\leq \ln\left[\frac{1}{2}\left(\frac{1}{\beta_t}+\beta_t\right)+\frac{1}{2}\left(\frac{1}{\beta_t}-\beta_t\right)||A\vec{x}_t-\vec{b}||_2\right].$$

Next, we have

$$\begin{aligned}
\sum_{i=1}^{n} x_i(\ln w_{(t+1),i}-\ln w_{t,i}) &= \sum_{i=1}^{n} x_i u_t(A_i)\ln\beta_t \\
&= (\ln\beta_t)\sum_{i=1}^{n} x_i\frac{(A_i-\vec{b})\cdot(\vec{b}-A\vec{x}_t)}{||\vec{b}-\vec{\lambda}_t||_2} \\
&= 0
\end{aligned}$$

since $\sum x_iA_i=\vec{b}$. Hence $I(\vec{x}||\vec{x}_{t+1})-I(\vec{x}||\vec{x}_t)$ is no more than

$$\ln\left[\frac{1}{2}\left(\frac{1}{\beta_t}+\beta_t\right)+\frac{1}{2}\left(\frac{1}{\beta_t}-\beta_t\right)||A\vec{x}_t-\vec{b}_t||_2\right].$$

Setting $E=||A\vec{x}_t-\vec{b}_t||_2$, we get

$$\begin{aligned}
I(\vec{x}||\vec{x}^{(t+1)})-I(\vec{x}||\vec{x}^{(t)}) &\leq \log\left[\frac{1}{2}\left(\frac{1}{\beta}+\beta\right)+\frac{1}{2}\left(\frac{1}{\beta}-\beta\right)E\right] \\
&= \log\left[\frac{1}{2}\left(\sqrt{\frac{1-E}{1+E}}+\sqrt{\frac{1+E}{1-E}}\right)+\frac{1}{2}\left(\sqrt{\frac{1-E}{1+E}}-\sqrt{\frac{1+E}{1-E}}\right)E\right] \\
&= \log\frac{1}{2}\left[(1+E)\sqrt{\frac{1-E}{1+E}}+(1-E)\sqrt{\frac{1+E}{1-E}}\right] \\
&= \log\frac{1}{2}\left[\sqrt{1+E}\sqrt{1-E}+\sqrt{1-E}\sqrt{1+E}\right] \\
&= \log\sqrt{1-E^2} \\
&\leq -\frac{1}{2}||A\vec{x}_t-\vec{b}||_2^2.
\end{aligned}$$

and $u_t : \mathbf{R}^n \to \mathbf{R}$ is defined by

$$u_t(\vec{y}) = \begin{cases} \frac{\vec{y}\cdot(\vec{b}-A\vec{x}_t)}{||\vec{b}-A\vec{x}_t||_2} & \text{if } A\vec{x}_t \neq \vec{b} \\ 1 & \text{otherwise.} \end{cases}$$

For each i and t, let

$$x_{t,i} = \frac{w_{t,i}}{\sum_{i=1}^n w_{t,i}}.$$

In the above, the columns $A_1, ..., A_n$ of the matrix correspond to the components $x_{t,1}, ..., x_{t,n}$ of the instances of the learning algorithm, each $\vec{x}_t$ corresponds to a $\vec{v}_t$, and $\vec{b}$ corresponds to a response ρ_t. Note that the base of the exponent is of a different form than that of the learning algorithm. This is due to the fact that we cannot guarantee that the exponent will be positive, and thus we use a different approximation for β^x. Optimizing β for the resulting bounds yields the above choice.

Note that if $||\vec{b} - A\vec{x}_t||_2$ is 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,[6] $A\vec{x}_t$ 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.

Also, note that we obtain the same sequence of estimates if we normalize $\vec{w}$ after each trial, as was done in the learning algorithm.

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 theorem is the basis for our analysis of this algorithm.

Theorem 14: *Let A be an $m\times n$ matrix and $\vec{b}$ be a $m\times 1$ column vector such that there exists $\vec{x} \in [0,1]^n, ||\vec{x}||_1 = 1$ with $A\vec{x} = \vec{b}$. Suppose further that for each $i, 1 \leq i \leq n, ||A_i||_2 < 1/2$. Then if $\vec{x}_1, \vec{x}_2, ...$ are defined as above,*

$$\sum_{t=1}^{\infty} ||A\vec{x}_t - \vec{b}||_2^2 \leq 2 \ln n.$$

Proof: First, note that due to the normalization of $\vec{x}$, we obtain the same sequence $\vec{x}_1, \vec{x}_2, ...$ if we let

$$u_t(\vec{y}) = \begin{cases} \frac{(\vec{y}-\vec{b})\cdot(\vec{b}-A\vec{x}_t)}{||\vec{b}-A\vec{x}_t||_2} & \text{if } A\vec{x}_t \neq \vec{b} \\ 1 & \text{otherwise.} \end{cases}$$

In this proof, we will assume that $\vec{w}_1, \vec{w}_2, ...$ are generated using this definition.

As in Theorem 8, we use $I(\vec{x}||\vec{x}_t)$ as a measure of progress. For each t, let $\vec{\lambda}_t = A\vec{x}_t$. Thus $\vec{\lambda}_t$ is the vector of right hand sides obtained by "plugging in" our proposed solution to the left hand sides of the input linear system.

Choose t. We have,

$$I(\vec{x}||\vec{x}_{t+1}) - I(\vec{x}||\vec{x}_t) = \left[\left(\ln \sum_{i=1}^n w_{(t+1),i}\right) - \left(\ln \sum_{i=1}^n w_{t,i}\right)\right] - \left(\sum_{i=1}^n x_i(\ln w_{(t+1),i} - \ln w_{t,i})\right).$$

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

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."

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

$$\sum_{t=1}^{n-1} (\lambda^{(t)} - \rho^{(t)})^2 = (cM)^2(n-1). \tag{5.2}$$

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 11, 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 11, the algorithm can be forced to have total loss of N in the second stage. Combining this with (5.2) yields the desired result. □

This result suggests that our algorithm is to be preferred to linear least squares 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 shares some characteristics with the learning algorithm treated in the previous sections. All vectors of this section are assumed to be column vectors.

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 $\vec{b}$ such that the L_2 norm of each of the columns of A and of $\vec{b}$ is less than $1/2$, produces a sequence $\vec{x}_1, \vec{x}_2, ...$ of $n \times 1$ column vectors.

Let $A_1, ..., A_n$ be the columns of A. The algorithm computes a sequence $\vec{w}_1, \vec{w}_2, ...$ of elements of $\mathbf{R}^n$ as follows. Let $w_{i,1} = 1$ for all i. For each t and i, let

$$w_{i,t+1} = w_{t,i} \beta_t^{u_t(A_i)}$$

where for each $t \in \mathbf{N}$,

$$\begin{aligned} \beta_t &= \left(\frac{1 + ||A\vec{x}_t - \vec{b}||_2}{1 - ||A\vec{x}_t - \vec{b}||_2} \right)^{\frac{1}{2}} \\ &= \exp(\tanh^{-1}(||A\vec{x}_t - \vec{b}||_2)) \end{aligned}$$

we have

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

Also,

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

so

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

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 8, 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] [BHL91].[5] (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 therefore omitted.

Corollary 12: *We have*

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

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 the number of variables n.

Theorem 13: *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*

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

[5] 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.

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. 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 vth 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 11: *We have*

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

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 $\mathrm{WA}(n, 1, k) \subseteq \mathrm{WA}(n, 1, \kappa)$. 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 $\vec{x}^{(1)}, ..., \vec{x}^{(l-k)}$ of the first stage are constructed as follows: $x_i^{(t)} = 1$ if $\mathrm{bit}(i, t) = 1$ and $i \leq 2^l$, 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 $\vec{\mu}$ as follows: if $i \leq 2^l$ and for each $t \leq l - k$, $\mathrm{bit}(i, t) = \rho^{(t)}$, then let $\mu_i = 2^{-k}$, and 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 $t, l - k < t < m$,

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

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 9 to get loss bounds for more general linear functions.

Theorem 10:

$$opt_{\mathrm{WA}}(n, M, \kappa, N) \le M^2 opt_{\mathrm{WA}}(n, 1, \kappa, N/M^2) \in O(M^2(\ln n - \kappa) + N)$$
$$opt_{\mathrm{LINEAR}}(n, M, c, N) \le (2cM)^2 opt_{\mathrm{WA}}(2n + 1, 1, 0, N/(2cM)^2) \in O((cM)^2 \ln n + N)$$

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

Choose n, M, 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 9 and Theorem 8.

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 ψ : 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

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

Note that for each i, ν_i is nonnegative, and that $\sum_i \nu_i = 1$. Choose $g \in$ LINEAR(n, M, c), $\vec{x} \in [0, M]^n$. Let $\vec{\mu}$ be the coefficient vector of g, $I^+ = \{i : \mu_i > 0\}$ and $I^- = \{i : \mu_i \le 0\}$. We have

$$\begin{aligned} \psi(g)(\phi(\vec{x})) &= \left(\sum_{i \in I^+} \frac{\mu_i(x_i + M)}{2cM}\right) + \left(\sum_{i \in I^-} \frac{-\mu_i(-x_i + M)}{2cM}\right) + \frac{1}{2}\left(1 - \frac{1}{c}\sum_i |\mu_i|\right) \\ &= \left(\frac{1}{2cM}\sum_{i=1}^{n} \mu_i x_i\right) + \left(\sum_i \frac{|\mu_i|}{2c}\right) + \frac{1}{2}\left(1 - \frac{1}{c}\sum_i |\mu_i|\right) \\ &= \frac{g(\vec{x})}{2cM} + 1/2. \end{aligned}$$

Thus there is a $2cM$-reduction from LINEAR(n, M, c) to WA$(2n + 1, 1, 0)$. The theorem now follows immediately from Theorem 9. The first bound can be proved by giving an M-reduction from WA(n, M, κ) to WA$(n, 1, \kappa)$ along the lines of the reduction given above. The details are omitted. □

3.5 Noise tolerance

Note that the smallest we can make the constant on the "noise term" (at the expense of the term depending on n and $H(\mu)$) by increasing δ 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 $k(\ln n - H(\mu)) + N(\mu)$ 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 [Hau88] [Lit88] [KLPV87] [PW90].

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 $\phi : X \rightarrow Y$, called an *instance transformation*, a function $\psi : \mathcal{F} \rightarrow \mathcal{G}$, 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.

Theorem 9: *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*

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

Proof: In Appendix A.

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 $f : [0, M]^n \rightarrow [0, M]$ such that there exists $\vec{\mu} \in [0,1]^n, ||\vec{\mu}||_1 = 1$ whose entropy is at least κ such that $f(\vec{x}) = \vec{\mu} \cdot \vec{x}$ 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 $f \in \mathrm{WA}(n, M, \kappa)$ such that

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

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

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

3.3 Choosing β

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_δ? Consider the upper bound for Δ_t given by the Inequality 3.2 for the case when $\rho_t = \vec{\mu}\cdot\vec{x}_t$:

$$\Delta_t \leq \ln(1+(\beta_t-1)\lambda_t') - \rho_t'\ln\beta_t.$$

Our above choice for β_t is obtained by minimizing this upper bound for Δ_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 β_t for the case when $\rho_t = \vec{\mu}\cdot\vec{x}_t$. Assume $fact(\beta_t, x_{t,i}) = \beta_t^{x'_{t,i}}$. 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 Δ_t, has exactly one minimum when $\beta_t \in [0,\infty]$. Denote the β_t at the minimum as $\beta_{t,opt}$. Now if we updated with $\beta_{t,opt}$ and fed $\vec{x}_t'$ to A_δ after the update was made, the algorithm would predict ρ_t. Thus with the optimum choice for β_t the algorithm is in some sense "corrective."

Since we have determined the choice for β_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_δ 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 β_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 β_t not too close to $\beta_{t,opt}$ and instead make a less drastic update.

3.4 Tuning δ

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

To get the remaining bounds we rewrite the second inequality

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

and apply Lemma 3, obtaining

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

The above immediately gives the first loss bound of the theorem:

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

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

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

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,

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

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

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

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 $\vec{\mu}_1 = ((1,0,\cdots,0),1)$ 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. However notice that the minimum in the loss bound is not achieved at the consistent vector $\vec{\mu}_1$.

Now, we wish to bound $|\ln\beta|$. First, let us assume that $\lambda_t \le \rho_t$. 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 5 and 6, we get that

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

By symmetry, when $\rho_t \le \lambda_t$, if we let $z = \lambda_t - \rho_t$, we obtain

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

Hence

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

Plugging into (3.3) yields the desired result. □

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

Theorem 8: *Choose $n, m \in \mathbf{N}$. Let $S = \langle(\vec{x}_t, \rho_t)\rangle_{1\le t\le m}$ be any sequence of m examples for $([0,1]^n, [0,1])$. Then for each $\delta > 0$,*

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

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

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

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

where $fact(\beta_t, x_{t,i}) \in [\beta_t^{\frac{x_{t,i}+\delta}{1+2\delta}}, 1+(\beta_t-1)\frac{x_{t,i}+\delta}{1+2\delta}]$ (any value in this range may be chosen by an implementor of the algorithm) and $\beta_t = \left(\frac{\rho_t+\delta}{\lambda_t+\delta}\right)\left(\frac{1-\lambda_t+\delta}{1-\rho_t+\delta}\right)$. [4] Note that since $\frac{x_{t,i}+\delta}{1+2\delta} \in (0,1)$, Lemma 2 assures that the interval in which $fact(\beta_t, x_{t,i})$ must lie has positive length.

As in [Lit89] 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 7: *Choose $\delta > 0$ and $n, t \in \mathbf{N}$. 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_δ and $\langle\lambda_t\rangle_{t\in\mathbf{N}}$ be the sequence of A_δ'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*

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

Proof: From the definition of Δ_t and $fact(\beta_t, x_{t,i})$ and from Lemma 2 it follows that

$$\begin{aligned}
\Delta_t &= \sum_{i=1}^n \mu_i \ln\frac{v_{t,i}}{v_{t+1,i}} \\
&= \ln(\sum_{i=1}^n v_{t,i} fact(\beta_t, x_{t,i})) + \sum_{i=1}^n \mu_i \ln\frac{1}{fact(\beta_t, x_{t,i})} \qquad (3.1)\\
&\leq \ln(\sum_{i=1}^n v_{t,i}(1+(\beta_t-1)x_{t,i}')) - \sum_{i=1}^n \mu_i x_{t,i}' \ln\beta_t \\
&= \ln(1+(\beta_t-1)\lambda_t') - \frac{\vec{\mu}\cdot\vec{x}_t+\delta}{1+2\delta}\ln\beta_t \\
&= \ln(1+(\beta_t-1)\lambda_t') - \rho_t'\ln\beta_t + \frac{\rho_t-\vec{\mu}\cdot\vec{x}_t}{1+2\delta}\ln\beta_t. \qquad (3.2)
\end{aligned}$$

Since β_t can be written as $\frac{\rho_t'}{\lambda_t'}\frac{1-\lambda_t'}{1-\rho_t'}$ we can rewrite the last expression as

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

leading to the first inequality of the lemma.

Next, we upper bound the last expression by using Lemma 1 and replacing the second term with its absolute value, obtaining:

$$\Delta_t \leq -2(\rho_t' - \lambda_t')^2 + \frac{|\rho_t - \vec{\mu}\cdot\vec{x}_t|\,|\ln\beta_t|}{1+2\delta}. \qquad (3.3)$$

[4]Note that if $\sigma(x) = (1+e^{-x})^{-1}$ is the usual sigmoid function, as δ goes to zero, β_t approaches $\exp(\sigma^{-1}(\lambda_t) - \sigma^{-1}(\rho_t))$.

Lemma 1 ([Kul67]): *For* $\lambda, \rho \in [0,1]$ $I((\rho, 1-\rho)||(\lambda, 1-\lambda)) \geq 2(\lambda-\rho)^2$.

The following series of lemmas give approximations for quantities arising in our analysis. The first three can be easily verified. The proofs of the last two are included in the appendix.

Lemma 2: *For all* $\beta > 0, x \in [0,1]$,

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

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

Lemma 3: *For all* $x, y \in \mathbf{R}$

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

Lemma 4: *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 5: *For all* z, δ *and* x *such that* $\delta > 0$, $0 < z \leq 1$ *and* $0 \leq x \leq 1 - z$,

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

Proof: In Appendix A.

Lemma 6: *For all* $\delta > 0$, *and* z *such that* $0 \leq z \leq 1$,

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

Proof: In Appendix A.

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 A_δ 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 δ is the more robust the algorithm is against noise, and, correspondingly, the more slowly the algorithm learns.

The Algorithm A_δ 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 δ:

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

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

$$\begin{aligned} ||\vec{x}||_1 &= \sum_{i=1}^{n} |x_i| \\ ||\vec{x}||_2 &= \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{x \cdot x}. \end{aligned}$$

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

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

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 ith component of $\bar{x}_t \in \mathbf{R}^n$ as $x_{t,i}$.

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 $\vec{\mu}, \vec{v} \in [0, 1]^n$ 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

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

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, $Y \subseteq \mathbf{R}$. 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 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 λ_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 *quadradic loss* 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 $S = \langle (x_t, \rho_t) \rangle_{1 \leq t \leq m}$ 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 three simple lemmas. The first is due to Kullback [Kul67].

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 [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.)

A variant of our learning algorithm leads to an algorithm for iteratively producing estimates $\vec{x}_1, \vec{x}_2, \cdots$ of the solution of the linear system $A\vec{x} = \vec{b}$ such that, if A^{-1} exists, the estimates converge to $A^{-1}\vec{b}$. Even if A is square and singular or is even non-square, if the system has a solution, then $A\vec{x}_1, A\vec{x}_2, ...$ converges to $\vec{b}$.

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. Further, our algorithm converges for all coefficient matrices. We know of no other algorithm which requires time linear in the number of nonzero entries in A which converges for all matrices.

The best bounds we can prove on the number of iterations required by our algorithm is significantly worse in some cases than the corresponding bounds for 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.

N is the total loss of the best fixed weight vector. It follows almost immediately from Mycielski's results that the Widrow-Hoff rule is within a constant of optimal for a closely related problem, where, instead of assuming that the hidden weight vector μ consists of nonnegative components summing to one, we assume that it has Euclidian length of at most one, and instead of choosing instances $\vec{x}_1, \vec{x}_2, ...$ from $[0,1]^n$, one assumes that the Euclidian length of the instances is at most 1. A more detailed comparison of our algorithm to the Widrow-Hoff rule including experimental comparisons is given in [CW91].

Our algorithms are motivated by the algorithms 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 linear functions defined on $\mathbf{R}^n$. 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] and the Widrow-Hoff rule [DH73], 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, 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

[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 $|\lambda_t - \rho_t|$.

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 λ_t. After predicting, the algorithm receives a real number ρ_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_δ 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 δ, 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 total loss of A_δ approaches $(\ln n)/2$ as δ 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 δ, A_δ 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_δ 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.[1] In particular, this implies that the total loss of A_δ 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 in [Hau88] [KLPV87] [Lit88] [PW90].

Mycielski [Myc88] gives worst case bounds on the total loss of the Widrow-Hoff rule (also sometimes called the delta rule) [WH60] [DH73]. However, instead of giving his bounds in terms of $\sum_{t=1}^{m}(\vec{\mu} \cdot \vec{x}_t - \rho_t)^2$, he gives bounds in terms of $\max_t(\vec{\mu} \cdot \vec{x}_t - \rho_t)^2$. His bounds, however, grow with the number of trials m. The focus of the research of this paper is to obtain bounds independent of m. Furthermore, it was shown in [CW91] that the worst-case total loss of the Widrow-Hoff rule in the setting of this paper is $\Omega(n + N)$, where, again,

[1]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 $\sum_{t=1}^{m}(\vec{\mu}' \cdot \vec{x}_t - \rho_t)^2 > 0$.

On-Line Learning of Linear Functions

Nicholas Littlestone*
Philip M. Long†
Manfred K. Warmuth‡

UCSC-CRL-91-29
October 1, 1991
Revised: October 30, 1991

Board of Studies in Computer and Information Sciences
University of California at Santa Cruz
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.

*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. Address: NEC Research Institute, 4 Independence Way, Princeton, NJ 08540. Email address: nickl@research.nj.nec.com.

†Supported by ONR grant N00014-91-J-1162. Email address: plong@saturn.ucsc.edu

‡Supported by ONR grant N00014-91-J-1162. Part of this work was done while employed by IIAS-SIS Fujitsu Limited in Numazu, Japan. Email address: manfred@mira.ucsc.edu.