Appeared in *Proceedings of the Seventh Yale Workshop on Adaptive and Learning Systems*, pp. 161–166, 1992.

Gain Adaptation Beats Least Squares?

Richard S. Sutton
GTE Laboratories Incorporated
Waltham, MA 02254
sutton@gte.com

Abstract

I present computational results suggesting that gain-adaptation algorithms based in part on connectionist learning methods may improve over least squares and other classical parameter-estimation methods for stochastic time-varying linear systems. The new algorithms are evaluated with respect to classical methods along three dimensions: asymptotic error, computational complexity, and required prior knowledge about the system. The new algorithms are all of the same order of complexity as LMS methods, $O(n)$, where n is the dimensionality of the system, whereas least-squares methods and the Kalman filter are $O(n^2)$. The new methods also improve over the Kalman filter in that they do not require a complete statistical model of how the system varies over time. In a simple computational experiment, the new methods are shown to produce asymptotic error levels near that of the optimal Kalman filter and significantly below those of least-squares and LMS methods. The new methods may perform better even than the Kalman filter if there is any error in the filter's model of how the system varies over time.

Introduction

Approximation of linear and nonlinear functions from examples is a central problem in many fields, including adaptive control and estimation, statistics, neural networks and machine learning. It is natural then to expect cross-fertilization, in which ideas developed in one field may also contribute to methods used in another. One topic of recent interest in connectionist learning is the automatic selection of learning-rate or gain parameters during learning (Jacobs, 1988; Lippmann & Lee, 1990; Sutton, 1992; Gluck, Glauthier & Sutton, 1992; cf. Kesten, 1958). Such dynamic-learning-rate (DLR) methods have been shown to speed convergence on a variety of learning tasks, particularly those with many irrelevant input features and in which the task is non-stationary (the correct solution changes over time). In this paper I present new linear parameter estimation algorithms that combine ideas from DLR methods with ideas from Kalman filtering and least-squares methods.

We will consider the classical problem of incrementally estimating the parameter vector $\theta(t)$ of the linear scalar system

$$y(t) = \theta(t)^T \phi(t) + \eta(t) \tag{1}$$

from observations of $y(t)$ and $\phi(t)$, $t = 0, 1, 2, \ldots$, where $\eta(t)$ is white noise and $\theta(t)$ is slowly varying over time. The observation noise $\eta(t)$ is mean-zero gaussian, with variance R. The parameter vector $\theta(t)$ drifts over time according to a simple random walk

$$\theta(t+1) = \theta(t) + d(t) \tag{2}$$

where $d(t)$, the *drift*, is a mean-zero, gaussian random vector with covariance matrix $Q = E\left\{d(t)d(t)^T\right\}$. We assume that $\eta(t)$, $d(t)$, and $\phi(t)$ are all independently chosen random variables. Popular estimation methods for this problem include the Kalman filter, least squares, and LMS (projection) methods.

We will consider three ways of evaluating algorithms: asymptotic error, computational complexity, and required prior knowledge of the system. The Kalman filter is indeed optimal in terms of asymptotic error, but requires accurate knowledge of Q, which is usually not available in practice. The closely related least-squares methods generally do not assume such knowledge, but neither do they perform as well as the Kalman filter. In addition, both of these methods are $O(n^2)$ in computational complexity, where n is the number of input signals (the dimensionality of $\phi(t)$ and $\theta(t)$). Their complexity alone rules out their use on large learning problems, where n may be thousands or tens of thousands. The LMS methods are only $O(n)$ in computational complexity, but perform significantly worse than both the Kalman filter and least-squares methods.

The three DLR methods I present here are also only $O(n)$ in computational complexity, but in our computational experiments they performed much better than both the LMS methods and least squares, as summarized in Figure 3. The best of the DLR methods (K1 and K2) in fact performed very nearly as well as the Kalman filter, despite their reduced complexity, and despite their lack of the special knowledge required by the Kalman filter.

The next section is a detailed specification of the new algorithms and the classical algorithms used in this study. The third section describes the specifics of the computational experiment and presents its results.

The Algorithms

This study considered seven algorithms: the Kalman filter, a least-squares method with covariance modification, the least-mean-square (LMS) algorithm, normalized LMS (NLMS), and three DLR algorithms called K1, K2, and IDBD. K1 and K2 are new to this paper whereas IDBD was introduced in (Sutton, 1992). This section completely describes the algorithms. They are all of the form

$$\hat{\theta}(t+1) = \hat{\theta}(t) + K(t)\Big[y(t) - \hat{\theta}(t)^T\phi(t)\Big] \tag{3}$$

where$\hat{\theta}(0) = \theta(0)$, and $K(t)$ is a gain vector that differs from algorithm to algorithm.

The Classical Algorithms

The LMS (projection) algorithm uses

$$K(t) = \frac{\mu\phi(t)}{\hat{R} + \hat{E}\left\{\phi(t)^T\phi(t)\right\}} \tag{4}$$

where $\mu > 0$ is a scalar constant, $\hat{R}$ is an estimate of $R = E\left\{\eta^2(t)\right\}$, and $\hat{E}\left\{\phi(t)^T\phi(t)\right\}$ is an estimate of the expected value of $\phi(t)^T\phi(t)$ (a scalar constant). In this study, the LMS algorithm used $\hat{E}\left\{\phi(t)^T\phi(t)\right\} = n$, where n is the dimensionality of $\phi(t)$, and all the algorithms used $\hat{R} = 1$. These estimates are exactly correct for the system used in the computational experiment.

The normalized LMS (NLMS) algorithm uses

$$K(t) = \frac{\mu\phi(t)}{\hat{R} + \phi(t)^T\phi(t)} \tag{5}$$

where $2 > \mu > 0$. This algorithm is normally slightly more efficient than LMS (see Goodwin & Sin, 1984)

We turn now to the least-squares and Kalman-filter algorithms. Given the form of the system time variation, (2), the appropriate least-squares method is least squares with covariance modification (e.g., see Goodwin & Sin, 1984). This method uses

$$K(t) = \frac{P(t)\phi(t)}{\hat{R} + \phi(t)^T P(t)\phi(t)} \tag{6}$$

where $P(t)$ is an $n \times n$ matrix computed recursively by

$$P(t+1) = P(t) - \frac{P(t)\phi(t)\phi(t)^T P(t)}{R + \phi(t)^T P(t)\phi(t)} + \hat{Q} \tag{7}$$

where $P(0) = 0$ and $\hat{Q}$ is an estimate of the drift covariance matrix $Q = E\left\{d(t)d(t)^T\right\}$. The algorithm I refer to as *LS* uses $\hat{Q} = \lambda I$, for $\lambda > 0$. This choice treats all input signals symmetrically, and reflects no prior knowledge of correlations among the drift variables.

If $\hat{Q} = Q$ and $\hat{R} = R$, then the algorithm given above is the Kalman filter (see, e.g., Goodwin & Sin, 1984). In this case the estimates $\hat{\theta}(t)$ can be shown to be optimal in the sense that

$$\hat{\theta}(t) = E\left\{\theta(t) \mid \mathcal{Y}(t-1)\right\}$$

where $\mathcal{Y}(t-1) = (y(t-1), y(t-2), \ldots, y(0), \phi(t-1), \phi(t-2), \ldots, \phi(0), \theta(0))$ is the history of all observations preceding time t plus initial conditions. Also in this case it can be shown that $P(t)$ is the estimation-error covariance matrix:

$$P(t) = E\left\{[\theta(t) - \hat{\theta}(t)][\theta(t) - \hat{\theta}(t)]^T \mid \mathcal{Y}(t-1)\right\} \tag{8}$$

However, in practice Q is never known exactly, and an approximation must be used. In this study, we will consider a family of Kalman-filter algorithms, parameterized by $\rho > 0$, which ranges from $\hat{Q} = Q$ at the $\rho = 0$ extreme, and $\hat{Q} = I$ at the $\rho = 1$ extreme:

$$\hat{Q} = (1-\rho)Q + \rho I \tag{9}$$

I refer to this in what follows as the *Kalman* algorithm.

The DLR Algorithms

If n is the dimension of the vectors $\theta(t)$ and $\phi(t)$, then any algorithm based on (3) requires memory and computation that is at least of $O(n)$. The LMS and NLMS algorithms are $O(n)$, whereas the Kalman and LS algorithms are $O(n^2)$ in both computation and memory. Clearly, no algorithm can be $O(n)$ that stores a full $n \times n$ matrix such as $P(t)$. However, we need not go as far as the LMS and NLMS algorithms, which have no memory other than $\hat{\theta}(t)$. The idea in algorithms K1 and K2 is to approximate $P(t)$ as a diagonal matrix, using memory only for the n diagonal entries. K1 and K2 use

$$K(t) = \frac{\hat{P}(t)\phi(t)}{\hat{R} + \phi(t)^T\hat{P}(t)\phi(t)} \tag{10}$$

where $\hat{P}(t)$ is a diagonal matrix with diagonal entries $\hat{p}_{ii}(t)$.

Algorithm K1 adapts the $\hat{p}_{ii}(t)$ by gradient descent in a corresponding set of parameters $\beta_i(t)$, $i = 1, \ldots, n$, where $\hat{p}_{ii}(t)$ is related to $\beta_i(t)$ by

$$\hat{p}_{ii}(t) = e^{\beta_i(t+1)} \tag{11}$$

K1 performs gradient descent in $\beta_i(t)$ rather than directly in $\hat{p}_{ii}(t)$ because then fixed-size steps (in $\beta_i(t)$) produce geometric steps in $\hat{p}_{ii}(t)$ (e.g., up or down by 10%), and because $\hat{p}_{ii}(t)$ is then assured of always being positive. Algorithm K1 approximates the gradient descent

$$\beta_i(t+1) = \beta_i(t) - \frac{1}{2}\mu\frac{\partial\delta^2(t)}{\partial\beta_i} \tag{12}$$

where $\delta(t)$ is the error, $\delta(t) = y(t) - \hat{\theta}(t)^T\phi(t)$, and $\mu > 0$ is a constant, the *meta-step-size* parameter. Algorithm K1's update equations were derived from (12) as shown in the appendix. They are

$$\beta_i(t+1) = \beta_i(t) + \mu\delta(t)\phi_i(t)h_i(t) \tag{13}$$

where

$$\beta_i(0) = \log\hat{R} \tag{14}$$

and where $h_i(t)$, $i = 1, \ldots, n$ is another set of modifiable parameters computed by

$$h_i(t+1) = \Big[h_i(t) + k_i(t)\delta(t)\Big]\Big[1 - k_i(t)\phi_i(t)\Big]^+ \tag{15}$$

where $k_i(t)$ is the ith component of $K(t)$ (see (10)) and $[x]^+$ is x if $x > 0$, else 0. Algorithm K1 thus requires memory for $2n$ additional numbers (n for $\beta_i(t)$ and n for $h_i(t)$) beyond the memory for the original n numbers for $\hat{\theta}(t)$. This may seem excessive, but of course it is far less than required by the LS and Kalman methods for storing the $n \times n$ matrix $P(t)$. I doubt that there is any way to effectively approximate the gradient descent (12) while storing fewer than $2n$ additional numbers, but the next algorithm, K2, shows that there are other ways of approximating $P(t)$ that require storing only n additional numbers.

The K2 algorithm approximates $p_{ii}(t)$ using $\beta_i(t)$ as K1 does, but it adapts the $\beta_i(t)$ in a totally different way. Note that $P(t)$ can be related to the expected squared error by

$$\begin{aligned}
E\left\{\delta^2(t)\right\} &= E\left\{\left[y(t) - \phi(t)^T\hat{\theta}(t)\right]^2\right\} \\
&= E\left\{\left[\phi(t)^T\theta(t) + \eta(t) - \phi(t)^T\hat{\theta}(t)\right]^2\right\} \\
&= E\left\{\left[\phi(t)^T[\theta(t) - \hat{\theta}(t)] + \eta(t)\right]^2\right\} \\
&= E\left\{\left[\phi(t)^T\epsilon(t) + \eta(t)\right]^2\right\}
\end{aligned}$$

(where $\epsilon(t) = \theta(t) - \hat{\theta}(t)$)

$$\begin{aligned}
&= E\left\{\left[\phi(t)^T\epsilon(t)\right]^2 + \eta(t)^2 + 2\eta(t)\phi(t)^T\epsilon(t)\right\} \\
&\approx E\left\{\phi(t)^T\epsilon(t)\epsilon(t)^T\phi(t)\right\} + R + 0 \\
&= \phi(t)^T E\left\{\epsilon(t)\epsilon(t)^T\right\}\phi(t) + R \\
&= R + \phi(t)^T P(t)\phi(t)
\end{aligned}$$

and for a diagonal $P(t)$ with diagonal entries $p_{ii}(t)$,

$$= R + \sum_i p_{ii}(t)\phi_i^2(t)$$

In other words, the $p_{ii}(t)$ are the coefficients of the best linear prediction of $\delta^2(t)$ from $\phi^2(t)$. An approximation of these coefficients can be formed by using any incremental regression method. The K2 algorithm uses an NLMS method to do this. That is, it uses (10), (11), and (14), and updates the $\beta_i(t)$ by

$$\beta_i(t+1) = \beta_i(t) + \frac{\mu\phi_i^2(t)}{1 + \sum_j \phi_j^4(t)}\left[\delta^2(t) - \hat{R} - \sum_i \hat{p}_{ii}(t)\phi_i^2(t)\right] \tag{16}$$

The K2 algorithm is strikingly similar in concept to that developed by Sanger, Matheus, and Sutton for a different purpose (Sanger, 1991; Sutton & Matheus, 1991; Sanger, Sutton & Matheus, 1992). Like K2, their method used the coefficients of a regression of the squared error onto the squared inputs to identify relevant inputs. In their method, however, the inputs identified as relevant in this way were not given higher learning rates, but rather were favored for being combined multiplicatively to produce higher order terms in a polynomial approximation of a nonlinear system.

Finally, the IDBD algorithm (Sutton, 1992) is the immediate ancester of the K1 algorithm. In short, K1 is to NLMS as IDBD is to LMS. The IDBD algorithm is defined by

$$k_i(t) = e^{\beta_i(t+1)}\phi_i(t) \tag{17}$$

with $\beta_i(t)$ defined by (13), with

$$\beta_i(0) = \log\frac{1}{n} \tag{18}$$

and

$$h_i(t+1) = h_i(t)\left[1 - k_i(t)\phi_i(t)\right]^+ + k_i(t)\delta(t) \tag{20}$$

The derivation of (13) and (20) from the gradient descent equation (12) is given in (Sutton, 1992). It is similar to that for K1 given in the appendix.

The Computational Experiment

A simple computational experiment was performed to assess the asymptotic performance of the seven algorithms. The dimension of the system was $n = 20$. The input signals $\phi_i(t)$ were independent gaussian random variables with mean zero and unit variance (i.e., $E\left\{\phi(t)\phi(t)^T\right\} = I$). The observation noise was also mean-zero gaussian with unit variance ($R = 1$). The first five components of the drift vector $d(t)$ were mean-zero gaussian with unit variance (uncorrelated), while the remaining 15 components were always zero (the drift covariance matrix $Q = E\left\{d(t)d(t)^T\right\}$ was all 0 except for the first 5 diagonal entries, which were 1). This means that only the first 5 components of θ changed over time, and thus asymptotically only the first 5 input signals were relevant. The other 15 inputs signals could and should be ignored for optimal tracking. All random variables ($\phi(t)$, $\eta(t)$, and $d(t)$) were chosen independently. The initial parameter vector was $\theta(0) = (0, \ldots, 0)^T$.

On this problem it suffices to perform a single long run for each algorithm and measure its asymptotic tracking performance. In this experiment, each algorithm was run for 20,000 time steps to get past any transients, and then for another 10,000 time steps. The square root of the mean-squared error (RMSE) between $y(t)$ and $\hat{\theta}(t)^T\phi(t)$ over the 10,000 time steps was used as the measure of the performance of the algorithm. The seed of the random number generator was reset at the beginning of each run, so that each algorithm experienced the exact same sequence of $y(t)$ and $\phi(t)$.

The algorithms were run at each of a range of values of their free parameter (μ, λ, or ρ). The results are plotted for each algorithm individually in separate panels of figure 1. Figure 2 combines the results from all seven algorithms by appropriately rescaling their free parameters into one range. For algorithms LMS, NLMS, K1, K2, and IDBD, results are shown for most of the feasible range of μ. For values slightly higher than those shown, these algorithms became unstable. The standard errors in all cases are smaller than the symbols marking the data points, and thus almost all perceptible differences are significant.

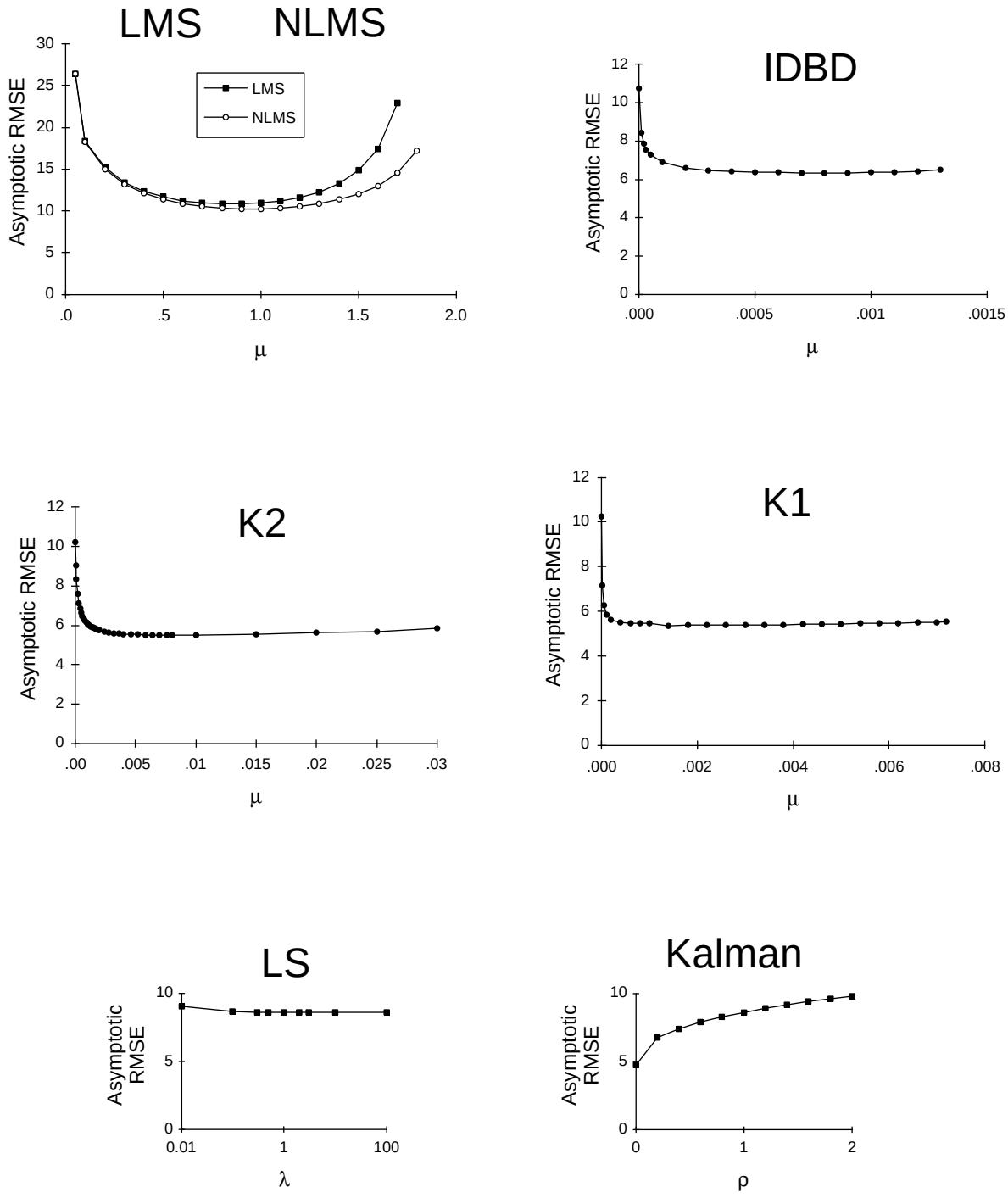

Figure 1: Detailed performance data on all algorithms as a function of their free parameter. Plotted on the vertical axis of each graph is the square root of the mean squared error averaged over the last 10,000 time steps of a long run. Each data point represents a different run using a different value for the algorithm's free parameter (μ, λ, or ρ).

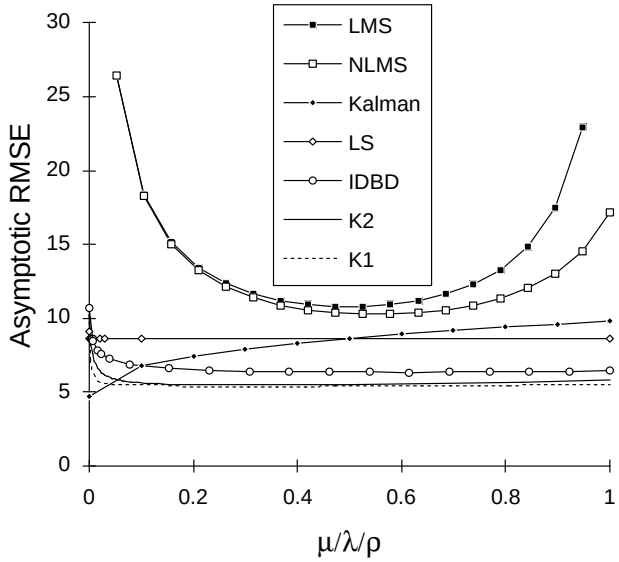

Figure 2: Performance comparison of all algorithms, combining all the data from figure 1 in one graph. Each algorithm's free parameter (μ, λ, or ρ) was rescaled such that the values for which data was obtained lay between 0 and 1.

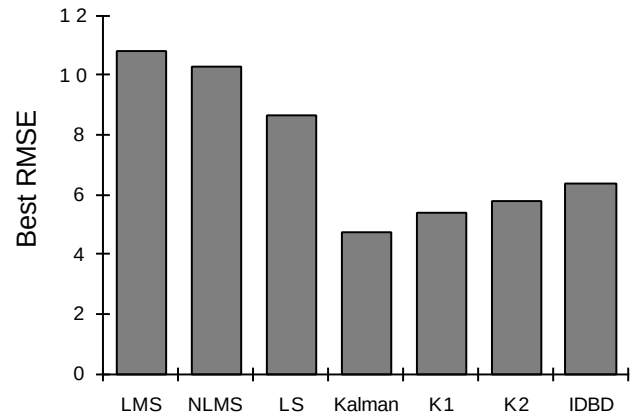

Figure 3: Summary performance comparison of all algorithms. The height of each column is the performance of the corresponding algorithm at the best value of its free parameter.

All the algorithms (except Kalman) are only weakly dependent upon the value of their free parameter when it is near its best value. It is thus reasonable to summarize the performance of the algorithms by taking their performance at their optimal parameter values, as is done in figure 3. The relative performance of the classical methods is consistent with expectations. The Kalman filter performed best, much better than the least-squares method (LS), which was much better that NLMS, which in turn was slightly better than LMS. The three DLR methods, however, all performed significantly better than least squares, and approached the performance of the Kalman filter. Moreover, inspection of figures 1 and 2 reveals that the Kalman algorithm performed better than the DLR algorithms only within a very narrow range of its free parameter ρ. If the Kalman algorithm's estimate $\hat{Q}$ was even slightly inaccurate, then K1, K2, and even IDBD performed better than the Kalman algorithm. This is only one experiment, and further work is needed, but these results do suggest that in practice the DLR methods may perform better than the LS and Kalman algorithms.

How could it be that the DLR algorithms outperform least-squares and Kalman filter methods? The DLR algorithms do not require or take advantage of any special prior knowledge of Q. In addition, the computational complexity of the new methods is at least an order of n less (see table 1 for a summary of the computational complexity of the algorithms). These limitations should put the DLR methods at a significant disadvantage relative to LS and Kalman methods. Because the DLR methods are $O(n)$, they can use only a crude, $O(n)$ model of the system's time variation, but unlike LS and Kalman, they adapt that model to the actual system. The LS and Kalman algorithms on the other hand use a fixed model of the system's time variation. If that model is exactly correct, then the Kalman filter is optimal, but if the model is even slightly in error, the Kalman and LS algorithms have no special status and can be beat even by $O(n)$ methods.

This perspective suggests that it may be possible to design an $O(n^2)$ method that is adaptive, that would approach the performance of the optimal Kalman filter and outperform all other methods. This is an interesting possibility for future research. We note, however, that for many problems such a method would be ruled out simply because of its $O(n^2)$ complexity.

Acknowledgements

The author wishes to thank Chris Matheus, Ron Williams, Oliver Selfridge, Chris Atkeson, and Judy Franklin for helpful discussions of these ideas, and additionally Hamid Benbrahim, Chris Matheus, Judy Franklin, Oliver Selfridge, and Marty Hiller for reading and commenting on an earlier draft of the paper.

Computational Complexity		
Algorithm	Memory	Adds & Mults
LMS	n	$4n$
NLMS	n	$6n$
K2	$2n$	$16n$
IDBD	$3n$	$13n$
K1	$3n$	$17n$
LS	$\frac{1}{2}n^2 + n$	$2.5n^2 + 8.5n$
Kalman	$\frac{1}{2}n^2 + n$	$2.5n^2 + 8.5n$

TABLE 1
Approximate computational complexity of the algorithms

References

Gluck, M.A., Glauthier, P.T., & Sutton, R.S. (1992) Adaptation of cue-specific learning rates in adaptive networks: Computational and psychological perspectives. *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society.*

Goodwin, G.C. & Sin, K.S. (1984) *Adaptive Filtering Prediction and Control.* Englewood Cliffs, NJ: Prentice-Hall, 1984.

Jacobs, R.A. (1988) Increased rates of convergence through learning rate adaptation. Neural Networks 1, 295–307.

Kesten, H. (1958) Accelerated stochastic approximation. *Annals of Mathematical Statistics 29*, 41–59.

Lee, Y. & Lippmann, R.P. (1990) Practical characteristics of neural network and conventional pattern classifiers on artificial and speech problems. In *Advances in Neural Information Processing Systems 2*, D.S. Touretzky, Ed., 168–177, Morgan Kaufmann.

Sanger T.D. (1991) Basis-function trees as a generalization of local variable selection methods for function approximation. In: *Advances in Neural Information Processing Systems 3*, Lippmann R.P., Moody J.E., Touretzky D.S., Eds., 700–706, Morgan Kaufmann.

Sanger, T.D., Sutton R.S., Matheus C.J. (1992) Iterative construction of sparse polynomial approximations. In *Advances in Neural Information Processing Systems 4*, Morgan Kaufmann.

Sutton, R.S. (1992) Adapting bias by gradient descent: An incremental version of delta-bar-delta. *Proceedings of the Tenth National Conference on Artificial Intelligence.*

Sutton R.S., Matheus C.J. (1991) Learning polynomial functions by feature construction. *Proc. Eighth Intl. Workshop on Machine Learning*, 208–212, Morgan Kaufmann.

Williams, R.J. & Zipser, D. (1989) Experimental analysis of the real-time recurrent learning algorithm. *Connection Science 1*, 87–111.

Appendix: Derivation of K1

This appendix presents a derivation of the K1 algorithm, (13) and (15), from (12). First we define some useful intermediate terms and derivatives:

$$\begin{aligned}\frac{\partial\delta^2(t)}{\partial\hat{\theta}_i(t)} &= 2\delta(t)\frac{\partial\delta(t)}{\partial\hat{\theta}_i(t)} = 2\delta(t)\frac{\partial}{\partial\hat{\theta}_i(t)}\Big[y(t)-\hat{\theta}(t)^T\phi(t)\Big] \\ &= -2\delta(t)\sum_j\frac{\partial\hat{\theta}_j(t)}{\partial\hat{\theta}_i(t)}\phi_j(t) = -2\delta(t)\phi_i(t) \qquad (A1)\end{aligned}$$

$$\begin{aligned}\frac{\partial\delta(t)}{\partial\beta_i} &= \frac{\partial}{\partial\beta_i}\Big[y(t)-\hat{\theta}(t)^T\phi(t)\Big] = -\sum_j\frac{\partial\hat{\theta}_j(t)}{\partial\beta_i}\phi_j(t) \\ &\approx -\frac{\partial\hat{\theta}_i(t)}{\partial\beta_i}\phi_i(t) \qquad (A2)\end{aligned}$$

The approximation above is reasonable in so far as the primary effect of changing β_i should be on the ith component of $\hat{\theta}$. That is, we assume

$$\frac{\partial\hat{\theta}_j(t)}{\partial\beta_i} \approx 0 \qquad for\ i \neq j \qquad (A3)$$

In these equations, the partial derivitive with respect to β_i without a time index should be interpretted as the derivative with respect to an infinitesimal change in β_i at all time steps. A similar technique is used in gradient-descent analyses of recurrent connectionist networks (c.f., e.g., Williams & Zipser, 1989). Some further intermediate computations will be needed:

$$D(t) \stackrel{def}{=} \hat{R} + \phi(t)^T\hat{P}(t)\phi(t) = \hat{R} + \sum_j e^{\beta_j(t+1)}\phi_j^2(t) \quad (A4)$$

$$k_i(t) \stackrel{def}{=} e^{\beta_i(t+1)}\phi_i(t)D^{-1}(t) \qquad (A5)$$

$$\frac{\partial D(t)}{\partial\beta_i} = \sum_j\frac{\partial e^{\beta_j(t+1)}}{\partial\beta_i}\phi_j^2(t) \approx e^{\beta_i(t+1)}\phi_i^2(t) \qquad (A6)$$

$$\frac{\partial D^{-1}(t)}{\partial\beta_i} = -D^{-2}(t)\frac{\partial D(t)}{\partial\beta_i} \approx -D^{-1}(t)k_i(t)\phi_i(t) \quad (A7)$$

$$\begin{aligned}\frac{\partial k_i(t)}{\partial\beta_i} &= \frac{\partial}{\partial\beta_i}\Big[e^{\beta_i(t+1)}\phi_i(t)D^{-1}(t)\Big] \\ &= \Big[\frac{\partial e^{\beta_i(t+1)}}{\partial\beta_i}\phi_i(t)D^{-1}(t) + e^{\beta_i(t+1)}\phi_i(t)\frac{\partial D^{-1}(t)}{\partial\beta_i}\Big] \\ &= \Big[k_i(t) + e^{\beta_i(t+1)}\phi_i(t)D^{-1}(t)k_i(t)\phi_i(t)\Big] \\ &= k_i(t)\Big[1-k_i(t)\phi_i(t)\Big] \qquad (A8)\end{aligned}$$

Now we are ready to expand (12):

$$\begin{aligned}\beta_i(t+1) &\approx \beta_i(t) - \frac{1}{2}\mu\frac{\partial\delta^2(t)}{\partial\beta_i} \\ &= \beta_i(t) - \frac{1}{2}\mu\sum_j\frac{\partial\delta^2(t)}{\partial\hat{\theta}_j(t)}\frac{\partial\hat{\theta}_j(t)}{\partial\beta_i} \\ &\approx \beta_i(t) - \frac{1}{2}\mu\frac{\partial\delta^2(t)}{\partial\hat{\theta}_i(t)}\frac{\partial\hat{\theta}_i(t)}{\partial\beta_i} \\ &= \beta_i(t) + \mu\delta(t)\phi_i(t)h_i(t) \qquad (A9)\end{aligned}$$

using (A1) and (A3), and where $h_i(t)$ is an approximation to $\frac{\partial\hat{\theta}_i(t)}{\partial\beta_i}$, derived as follows

$$\begin{aligned}h_i(t+1) &\approx \frac{\partial\hat{\theta}_i(t+1)}{\partial\beta_i} \\ &= \frac{\partial}{\partial\beta_i}\Big[\hat{\theta}_i(t) + k_i(t)\delta(t)\Big] \\ &\approx h_i(t) + k_i(t)\frac{\partial\delta(t)}{\partial\beta_i} + \delta(t)\frac{\partial k_i(t)}{\partial\beta_i} \\ &\approx h_i(t) - k_i(t)h_i(t)\phi_i(t) + \delta(t)k_i(t)\Big[1-k_i(t)\phi_i(t)\Big] \\ &= h_i(t)\Big[1-k_i(t)\phi_i(t)\Big] + \delta(t)k_i(t)\Big[1-k_i(t)\phi_i(t)\Big] \\ &= \Big[h_i(t)+\delta(t)k_i(t)\Big]\Big[1-k_i(t)\phi_i(t)\Big] \qquad (A10)\end{aligned}$$

using (3), (A2), and (A8). After adding a positive-bounding operation, this is the original update rule for $h_i(t)$, (15), while the derived update (A9) for $\beta_i(t)$ is the same as (13).

Q.E.D.

This is a re-typeset version of the original publication. Some line and page breaks were changed slightly, and minor corrections were made to equations (8), (15), (17) and (A10). The title in the first reference was corrected. [8/15/95]