
PERGAMON

AVAILABLE AT
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Neural Networks 16 (2003) 735–744

Neural Networks

www.elsevier.com/locate/neunet

2003 Special issue

Statistical efficiency of adaptive algorithms

Bernard Widrow*, Max Kamenetsky

ISL, Department of Electrical Engineering, Stanford University, Rm. 273, Packard Electrical Bldg. 350 Serra Mall, Stanford, CA 94305, USA

Abstract

The statistical efficiency of a learning algorithm applied to the adaptation of a given set of variable weights is defined as the ratio of the quality of the converged solution to the amount of data used in training the weights. Statistical efficiency is computed by averaging over an ensemble of learning experiences. A high quality solution is very close to optimal, while a low quality solution corresponds to noisy weights and less than optimal performance.

In this work, two gradient descent adaptive algorithms are compared, the LMS algorithm and the LMS/Newton algorithm. LMS is simple and practical, and is used in many applications worldwide. LMS/Newton is based on Newton's method and the LMS algorithm. LMS/Newton is optimal in the least squares sense. It maximizes the quality of its adaptive solution while minimizing the use of training data. Many least squares adaptive algorithms have been devised over the years, but no other least squares algorithm can give better performance, on average, than LMS/Newton.

LMS is easily implemented, but LMS/Newton, although of great mathematical interest, cannot be implemented in most practical applications. Because of its optimality, LMS/Newton serves as a benchmark for all least squares adaptive algorithms. The performances of LMS and LMS/Newton are compared, and it is found that under many circumstances, both algorithms provide equal performance. For example, when both algorithms are tested with statistically nonstationary input signals, their average performances are equal. When adapting with stationary input signals and with random initial conditions, their respective learning times are on average equal. However, under worst-case initial conditions, the learning time of LMS can be much greater than that of LMS/Newton, and this is the principal disadvantage of the LMS algorithm. But the strong points of LMS are ease of implementation and optimal performance under important practical conditions. For these reasons, the LMS algorithm has enjoyed very widespread application. It is used in almost every modem for channel equalization and echo cancelling. Furthermore, it is related to the famous backpropagation algorithm used for training neural networks.

Keywords: LMS; Newton algorithm; Least squares; Efficiency

1. Introduction

Learning systems take on many forms. Of special interest here are the adaptive linear combiner of Fig. 1 and the adaptive transversal filter of Fig. 2. The linear combiner is the basic building block of almost all neural networks and adaptive filters. The linear combiner and the adaptive filter have found very wide application in practice (Widrow, Mantey, Griffiths, & Goode, 1967; Widrow, Glover, McCool, Kaunitz, Williams, Hearn, Zeidler, Dong & Goodlin, 1975; Widrow & Stearns, 1985; Widrow & Walach, 1996; Ghogho, Ibnkahla, & Bershad, 1998; Proakis, 2001). For example, the present-day Internet would not exist without adaptive filters in every modem.

A learning system is generally characterized as an operator on signals, images, sounds, etc. that has adjustable parameters and that has a mechanism, an adaptive algorithm, for automatically adjusting the parameters in order to optimize the operator's performance. The adjustable parameters in Figs. 1 and 2 are adjustable weights, indicated by circles with arrows through them. The input signals are stochastic, and information obtained from the inputs is used by the adaptive algorithm to adjust the weights. The algorithm is thus a consumer of data. An efficient algorithm minimizes the usage of data while maximizing the quality of the solution, i.e. achieving parameter adjustments close to optimum. Minimizing data usage and maximizing solution quality are generally antagonistic. Minimizing data usage corresponds to fast adaptive convergence. But fast convergence based on a small amount of data could lead to a solution of poor quality. This tradeoff is present in all learning systems.

* Corresponding author.
E-mail addresses: widrow@stanford.edu (B. Widrow); maxk@stanford.edu (M. Kamenetsky).

0893-6080/03/$ - see front matter © 2003 Elsevier Science Ltd. All rights reserved.
doi:10.1016/S0893-6080(03)00126-6

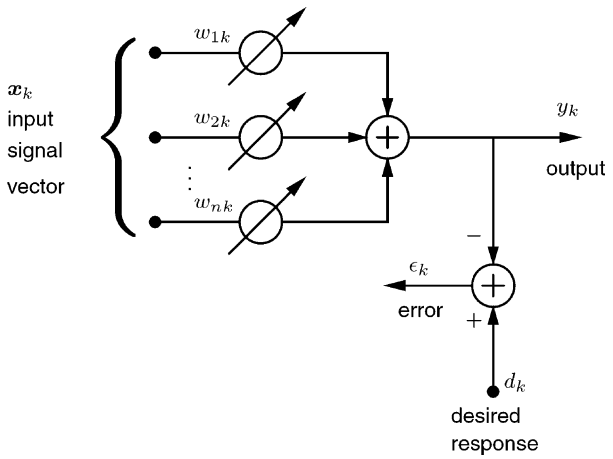


Fig. 1. Adaptive linear combiner.

2. The LMS and LMS/Newton algorithms

When referring to the linear combiner of Fig. 1, the set of weights is designated by the weight vector

$$\boldsymbol{w}_k = \begin{bmatrix} w_{1k} \\ w_{2k} \\ \vdots \\ w_{nk} \end{bmatrix}, \tag{1}$$

and the set of input signals to the weights is the input vector

$$\boldsymbol{x}_k = \begin{bmatrix} x_{1k} \\ x_{2k} \\ \vdots \\ x_{nk} \end{bmatrix}. \tag{2}$$

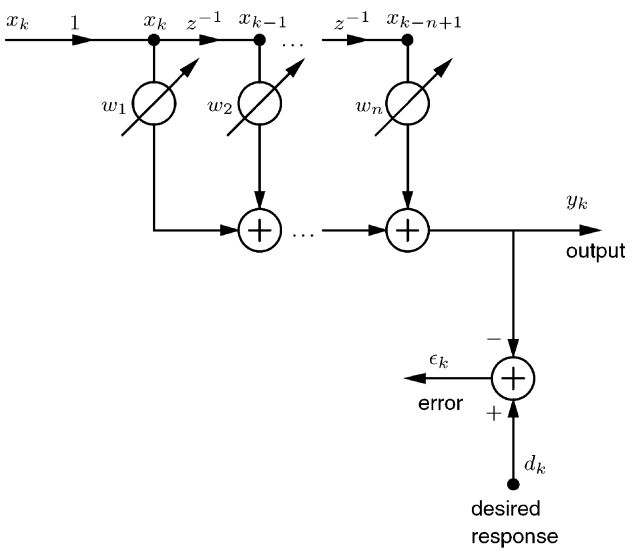


Fig. 2. Adaptive transversal digital filter.

The output signal y_k is the inner product of $\boldsymbol{x}_k$ and $\boldsymbol{w}_k$:

$$y_k = x_k^{\mathrm{T}} w_k = w_k^{\mathrm{T}} x_k. \tag{3}$$

The subscripts k are a time index.

The desired response d_k is an input training signal that is obtained in practice from the physical context of the application. Many examples of how desired responses may be obtained are shown in Widrow and Stearns (1985) for a variety of practical applications, such as prediction, noise cancelling, sensor array processing, and so forth. The error signal in Fig. 1 is $\boldsymbol{\epsilon}_k$, the difference between the desired response and the actual output signal:

$$\boldsymbol{\epsilon}_k = d_k - y_k. \tag{4}$$

Certain statistical properties of the inputs to the linear combiner are of importance. Assuming that the input and the desired response are stationary, the input autocorrelation matrix, designated by $\mathbf{R}$, is defined as

$$\mathbf{R} \triangleq E[\mathbf{x}_k \mathbf{x}_k^{\mathrm{T}}], \tag{5}$$

and the crosscorrelation vector between the input $\mathbf{x}_k$ and the desired response d_k is defined as

$$\mathbf{p} \triangleq E[d_k \mathbf{x}_k]. \tag{6}$$

For a given weight vector, with stationary input $\mathbf{x}_k$ and desired response d_k flowing into the linear combiner, the mean square error (MSE) is

$$\xi \triangleq E[\boldsymbol{\epsilon}_k^2] = E[d_k^2] - 2\mathbf{p}^{\mathrm{T}}\boldsymbol{w} + \boldsymbol{w}^{\mathrm{T}}\mathbf{R}\mathbf{w}, \tag{7}$$

where we dropped the subscript k from the vector w because we do not yet wish to adjust the weights. The MSE is thus a quadratic function of the weights. If there were only two weights, the MSE could be plotted as in Fig. 3.

With many weights, the surface, known as the performance surface, is a hyperparaboloid. The gradient of this

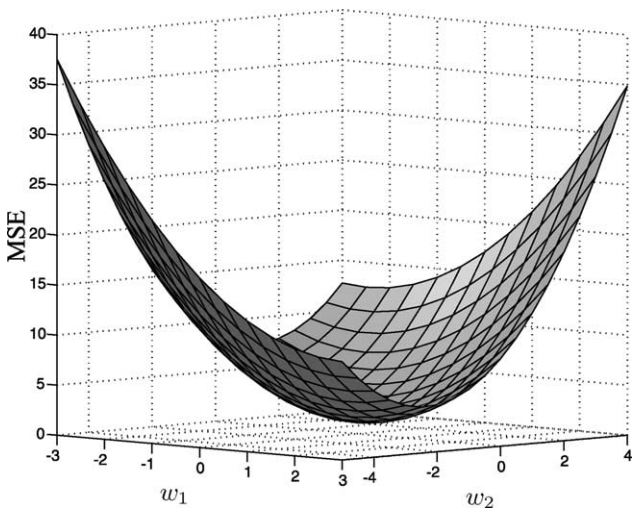


Fig. 3. Sample MSE for a two-weight system.

surface, is a vector obtained by differentiation of Eq. (7):

$$\nabla \triangleq \begin{bmatrix} \frac{\partial \xi}{\partial w_1} \\ \vdots \\ \frac{\partial \xi}{\partial w_n} \end{bmatrix} = -2\mathbf{p} + 2\mathbf{R}\mathbf{w}. \tag{8}$$

Assuming R is not singular, the optimal solution $\boldsymbol{w}^*$ is found by setting the gradient to zero:

$$\boldsymbol{w}^* = \mathbf{R}^{-1}\mathbf{p}. \tag{9}$$

This weight vector is the best linear least squares solution and is commonly known as the Wiener solution. The MSE corresponding to the Wiener solution is designated by $\xi_{\min}$.

The Wiener solution is of great theoretical interest, but it cannot be used directly in many practical circumstances because one would not know the statistics $\mathbf{R}$ and $\mathbf{p}$. Input data samples are available, and they can be used on an iterative basis as they arrive for the adjustment or adaptation of the weights. The weight adjustment process that is the simplest and most widely used in the world today is the Widrow-Hoff LMS algorithm, derived in 1959 (Widrow & Hoff, 1960). This algorithm is based on the method of steepest descent, using instantaneous gradients. It is interesting to note that the backpropagation algorithm of Werbos (1974) also utilizes the method of steepest descent using instantaneous gradients in adapting the weights of a neural network. Backpropagation is a remarkable generalization of the LMS algorithm. It is the most widely used algorithm for adapting neural networks.

Steepest descent can be written as

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \mu(-\nabla_k). \tag{10}$$

The next weight vector, $\boldsymbol{w}_{k+1}$, equals the present weight vector, $\boldsymbol{w}_k$, plus a change which is proportional to the negative gradient. The proportionality constant is μ, and this is a design parameter that controls stability and rate of convergence. The LMS algorithm is obtained from Eqs. (10) and (4) as

$$\begin{cases} \boldsymbol{w}_{k+1} = \boldsymbol{w}_k + 2\mu\epsilon_k\mathbf{x}_k, \\ \epsilon_k = d_k - \boldsymbol{x}_k^{\mathrm{T}}\boldsymbol{w}_k. \end{cases} \tag{11}$$

The gradient for each iteration is instantaneous and is given by $-2\epsilon_k\mathbf{x}_k$, as shown in Widrow and Stearns (1985).

When using steepest descent to find the minimum of a quadratic function of the weights, the weights progress geometrically toward the Wiener solution. In fact, each weight converges toward its Wiener value with a progression that is a sum of geometric progressions. Each of the geometric progressions has an individual 'time constant' (the unit of time is the iteration cycle). It has been shown (Widrow & Stearns, 1985) that there are as many distinct time constants as there are distinct eigenvalues of the $\mathbf{R}$ matrix. These time constants depend on the eigenvalues of $\mathbf{R}$ and correspond to natural modes of the adaptive algorithm. The relative amplitudes of the modes are different from one weight to another and depend on the initial conditions of the weight vector, i.e. its initial setting.

Since one rarely has a priori knowledge of the orientation of the initial weight vector setting with respect to the eigenvectors of the $\mathbf{R}$ matrix, it is difficult to predict the relative amplitudes of the modes and therefore difficult to predict the rate of convergence of the LMS algorithm. In spite of this drawback, LMS is very widely used.

A more predictable algorithm is Newton's method. Assuming that $\mathbf{R}$ is not singular, it can be written as

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \mu\lambda_{\mathrm{ave}}\mathbf{R}^{-1}(\nabla_k). \tag{12}$$

Here, the negative gradient is premultiplied by $\mu\lambda_{\mathrm{ave}}\mathbf{R}^{-1}$, where λ_{ave} is the average of the eigenvalues of $\mathbf{R}$. Based on Eq. (12) and using the instantaneous gradient $-2\epsilon_k\mathbf{x}_k$, the LMS/Newton algorithm can be written as

$$\begin{cases} \boldsymbol{w}_{k+1} = \boldsymbol{w}_k + 2\mu\lambda_{\mathrm{ave}}\mathbf{R}^{-1}\epsilon_k\mathbf{x}_k, \\ \epsilon_k = d_k - \mathbf{x}_k^{\mathrm{T}}\boldsymbol{w}_k. \end{cases} \tag{13}$$

With Newton's method, there is only one natural mode, corresponding to one time constant. The learning rate of Newton's method is independent of the weight vector's initial conditions. It is interesting to note that Eq. (13) is identical to Eq. (11) when all of the eigenvalues of $\mathbf{R}$ are equal.

So, the good news about LMS/Newton is that its rate of convergence is predictable and does not depend on initial conditions. The bad news is that one cannot implement this algorithm in practice because $\mathbf{R}^{-1}$ is generally unknown. LMS, based on steepest descent, has disadvantages, but it is simple and easy to implement. Also, as will be shown, it performs equivalently to LMS/Newton under many important conditions.

3. Learning with a finite set of data samples

The LMS/Newton algorithm is not only very predictable in its convergence behavior, but it is also highly efficient in its use of input data. It will be shown that in the least squares sense, no other algorithm can be more efficient in its data usage than LMS/Newton.

In order to study the question of efficiency for the LMS/Newton algorithm, it is useful to contemplate training the linear combiner with a finite number of data samples. One data sample consists of an $\mathbf{x}$ vector and its associated desired response. Assume that a set of N training input samples and associated desired responses is drawn from a given distribution. Define a matrix $\mathbf{X}$ for these training samples as the set of N $\mathbf{x}$-vectors:

$$\boldsymbol{X} \triangleq [\,\boldsymbol{x}_1 \quad \boldsymbol{x}_2 \quad \ldots \quad \boldsymbol{x}_N\,]^{\mathrm{T}}. \tag{14}$$

Define a desired response vector for these samples as

$$\mathbf{d} \triangleq [\, d_1 \quad d_2 \quad \ldots \quad d_N \,]^{\mathrm{T}}. \tag{15}$$

For a given weight vector w, the set of output responses is

$$\mathbf{y} \triangleq [\, y_1 \quad y_2 \quad \ldots \quad y_N \,]^{\mathrm{T}} = \mathbf{X}\mathbf{w}. \tag{16}$$

Define an error vector for these samples as

$$\boldsymbol{\epsilon} \triangleq [\, \epsilon_1 \quad \epsilon_2 \quad \ldots \quad \epsilon_N \,]^{\mathrm{T}} = \mathbf{d} - \mathbf{y}. \tag{17}$$

The objective is to find a set of weights that will minimize the sum of the squares of the errors for the training sample, i.e. minimize $\boldsymbol{\epsilon}^{\mathrm{T}}\boldsymbol{\epsilon}$. Assuming $\mathbf{X}$ is full rank, one obtains the optimal least squares solution either by differentiating $\boldsymbol{\epsilon}^{\mathrm{T}}\boldsymbol{\epsilon}$ with respect to $\boldsymbol{w}$ and setting the result to zero or by invoking the orthogonality condition (Kailath, Sayed, & Hassibi, 2000; Widrow & Stearns, 1985). The resulting least squares solution is

$$\boldsymbol{w}_{\mathrm{LS}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{d}. \tag{18}$$

One could directly calculate this solution from the data, or the same result could be obtained by training the weights with the LMS algorithm or any other least squares algorithm, recirculating the training data over and over until the weights converge and stabilize. It should be noted that if the number N were increased without bound, the finite least squares solution of Eq. (18) would converge in the mean-square sense to the true Wiener solution for the given distribution of training samples and desired responses.

In practice, one would have only a finite number of data samples available for training. The question is, given N training samples, how well does the optimal solution of Eq. (18) perform compared to the Wiener solution when operating on an infinite set of samples drawn from the same distribution? The true Wiener solution yields the minimum MSE. The least squares solution based on N training samples produces more MSE than the true Wiener solution. There is an excess MSE. If a different set of N data samples were randomly selected from the same distribution, a different optimal solution (18) would result, with a different excess MSE. Given an ensemble of randomly selected data sets, each with N samples of data, there would be an ensemble of optimal solutions (18) with an ensemble of excess MSEs. One could take an ensemble average of the excess MSEs and normalize with respect to the minimum MSE of the true Wiener solution. This dimensionless ratio is called the misadjustment due to training with N data samples:

$$M \triangleq \frac{\text{(average excess MSE)}}{\text{(minimum MSE)}}$$

$$= \frac{\text{(number of weights)}}{\text{(number of training samples)}} = \frac{n}{N}. \tag{19}$$

A derivation of this result is given in Widrow and Kamenetsky (2003) and Widrow, McCool, Larimore, and Johnson (1976). It was first reported by Widrow and Hoff at a WESCON conference in 1960 (Widrow & Hoff, 1960). Years of experience with adaptive filters and neural networks lead one to accept a misadjustment of 10% as a reasonable design value. This gives a performance that is only 10% worse than that of the optimal Wiener solution. When training a linear combiner, a 10% misadjustment is obtained when the number of training samples is equal to 10 times the number of weights. When training a neural network having only one neuron, the number of training patterns would be equal to 10 times the number of weights. When training an adaptive filter with a steady flow of stationary input data, the parameter μ should be chosen so that the training time or convergence time (several time constants of the learning curve) would be equal to 10 times the length of its impulse response in order to have a misadjustment of 10%. There is no similar law for multilayered neural networks trained with backpropagation, but one may speculate that for a network with many inputs and a single output, regardless of the number of neurons and layers, training with a number of patterns equal to 10 times the number of inputs should give good performance.

4. Learning with LMS/Newton

Consider the linear combiner, with stationary input data flowing in real time. Starting from an initial (non-optimal) weight vector and adapting with the LMS/Newton algorithm, the weight vector will geometrically (exponentially) relax toward the Wiener solution, with noise superposed. The noise originates from adapting with instantaneous gradients. (Estimating the gradient with a single data sample produces a noisy gradient). As the weights relax toward the Wiener solution, the MSE also relaxes exponentially toward the level of the minimum mean square error $\xi_{\min}$, with noise superposed. Because of gradient noise, the weights do not converge on the Wiener solution, but undergo Brownian motion about it. The MSE is almost always greater than $\xi_{\min}$ and never goes below it, because of the noise in the weights while adapting. A plot of MSE versus number of iterations is shown in Fig. 4, where ξ_{asy} is the asymptotic MSE, and ξ_{excess} is the difference between it and the minimum MSE $\xi_{\min}$. This type of plot is called the 'learning curve'. Note that, in general, $\xi_{\mathrm{excess}} > 0$ because the misadjustment is not zero, and thus, the asymptotic MSE ξ_{asy} is greater than $\xi_{\min}$, which is reflected in Fig. 4.

After gross adaptive transients have died out (after about four exponential time constants), the weights are in a steady state Brownian motion and the MSE hovers randomly above $\xi_{\min}$, exhibiting 'excess mean square error'. The misadjustment for this situation is defined as the average of the excess mean square error divided by $\xi_{\min}$:

$$M \triangleq \frac{\text{(average excess MSE)}}{\text{(minimum MSE)}}. \tag{20}$$

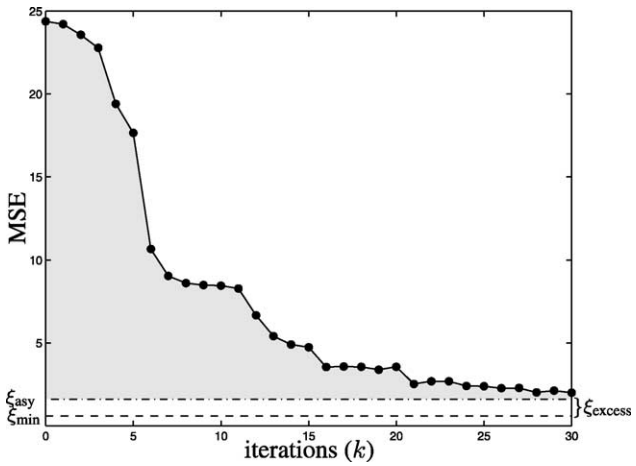


Fig. 4. Sample learning curve with gradient noise.

The bigger one makes μ, the faster the algorithm converges, and the greater is the misadjustment. It has been shown (Widrow & Kamenetsky, 2003; Widrow & Stearns, 1985; Widrow & Walach, 1984) that a good approximation for the misadjustment is

$$M = \mu \mathrm{Tr}(\boldsymbol{R}) = \frac{n}{4\tau_{\mathrm{MSE}}}, \tag{21}$$

where $\mathrm{Tr}(\mathbf{R})$ is the trace of $\mathbf{R}$, and τ_{MSE} is the time constant of the mean square error learning curve. The unit of time is the iteration cycle. Time is marked by number of iteration cycles, and since a new input training sample is used with each iteration cycle, time can equivalently be marked by number of training samples used.

5. Optimality of LMS/Newton in a stationary environment

The LMS/Newton algorithm exponentially weights its input data over time as it establishes its weight values. The settling time of the adaptive process is of the order of four time constants of the MSE learning curve. At any moment, the weights are determined by adaptation that has taken place over essentially the last four time constants worth of data. Thus, in a steady flow situation, the training data 'consumed' or 'absorbed' at any time by the LMS/Newton algorithm is essentially comprised of the most recent $4\tau_{\mathrm{MSE}}$ samples. From Eq. (21), the misadjustment of the LMS/Newton algorithm can therefore be expressed as

$$M = \frac{n}{4\tau_{\mathrm{MSE}}} = \frac{(\text{number of weights})}{(\text{number of independent training samples})}. \tag{22}$$

When learning with a finite set of data samples, the optimal weight vector is the best least squares solution for that set of samples, and it is often called the 'exact least squares solution'. This solution, given by Eq. (18), makes the best use of the finite number of data samples, in the least squares sense. All of the data are weighted equally in affecting the solution. This solution will of course vary from one set of samples to another, and it has a misadjustment of

$$M = \frac{(\text{number of weights})}{(\text{number of independent training samples})}. \tag{23}$$

For the same consumption of data, it is apparent that LMS/Newton and exact least squares yield the same misadjustment. Although we are comparing 'apples with oranges' by comparing a steady flow algorithm with an algorithm that learns with a finite number of data samples, we nevertheless find that LMS/Newton is as efficient as exact least squares when we relate the quality of the weight-vector solution to the amount of data used in obtaining it. Since the exact least squares solution makes optimal use of the data, so does LMS/Newton.

6. Transient learning with stationary input data

The LMS/Newton algorithm makes optimal use of its training data. Its learning curve is exponential with a single time constant. For practical purposes, its convergence time is of the order of four time constants, although in principle, convergence would take forever. Fast convergence is desirable because, during the initial learning transient, the MSE is excessive. The LMS algorithm has a learning curve which is a sum of exponentials. But, when all of the eigenvalues of the $\mathbf{R}$ matrix are equal, LMS has a single exponential learning curve that is identical to that of LMS/Newton; and LMS is therefore optimal. Generally, the eigenvalues are not equal, and LMS has a different kind of learning curve than LMS/Newton.

After learning transients die out, the steady state misadjustment for LMS/Newton is

$$M = \mu \mathrm{Tr}(\mathbf{R}), \tag{24}$$

and the steady state misadjustment for LMS is

$$M = \mu \mathrm{Tr}(\mathbf{R}). \tag{25}$$

The derivations of Eqs. (24) and (25) are given in Widrow and Kamenetsky (2003) and Widrow and Stearns (1985).

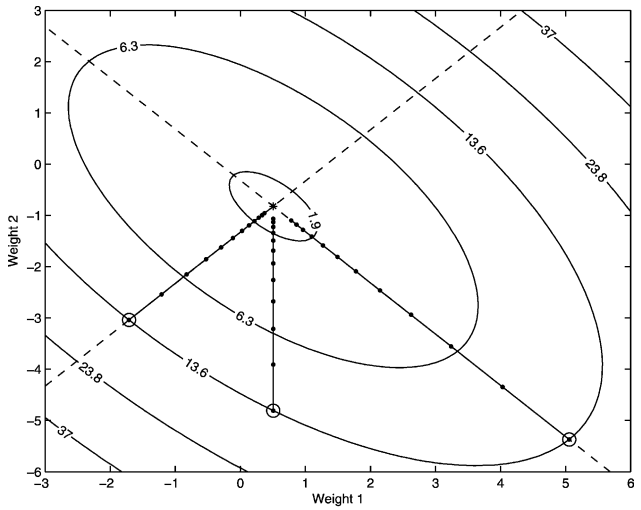
These misadjustment formulas are approximations of the true misadjustment. It should be noted that more precise formulas for misadjustment have been derived by, among others, Butterweck (2001), Dabeer and Masry (2002), Widrow and Walach (1996, App. A), and Yousef and Sayed (2001). However, the formulas in Eqs. (24) and (25) have been used for many years and have proven to be excellent approximations for small values of M (less than 25%).

A fair and reasonable way to compare two algorithms is to adjust their rates of convergence so that they would have the same steady state misadjustment. One could always make an algorithm 'look good' with fast adaptation, but this

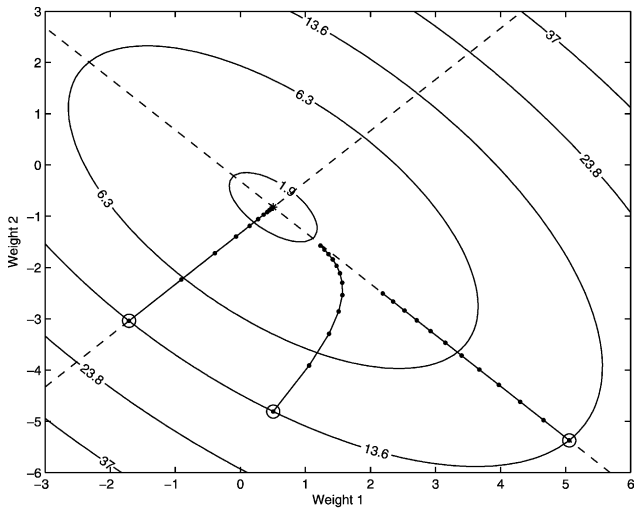
causes excessive misadjustment. To compare LMS with LMS/Newton, their μ-parameters should be set to the same value.

Fig. 5 shows plan views of a quadratic MSE surface, Fig. 5(a) indicating adaptive steps for Newton's method and Fig. 5(b) showing corresponding steps for the method of steepest descent with equivalent initial conditions. These steps correspond to three adaptive transient experiments, each starting from a different point on the same contour of constant MSE and operating with the same value of μ. The steps using Newton's method are always directed toward the bottom of the quadratic bowl, whereas those of steepest descent follow the local gradient, orthogonal to the contours of constant MSE.

Fig. 6 shows learning curves corresponding to the adaptive steps illustrated in Fig. 5. All three learning curves



(a)



(b)

Fig. 5. Illustration of Newton's method versus steepest descent: (a) Newton's method, (b) steepest descent. The Wiener solution is indicated by $*$. The three initial conditions are indicated by ○.

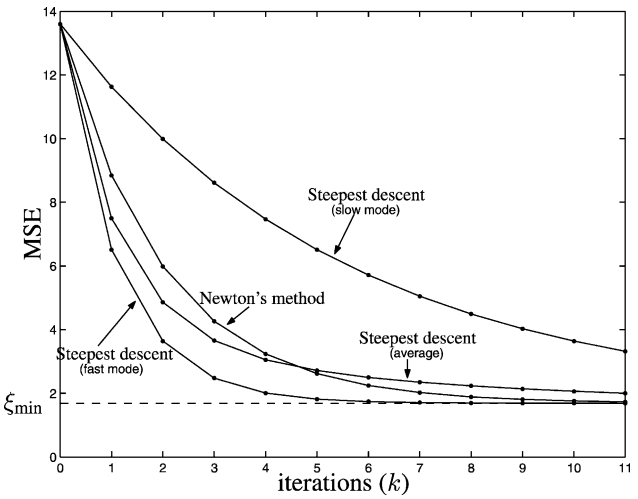


Fig. 6. Steepest descent and Newton's method learning curves.

derived from Newton's method are identical, since the initial starting conditions are located on the same constant MSE contour, and all three time constants are the same. Fig. 6 shows all three learning curves as a single curve labeled 'Newton's method'. The three steepest descent curves are distinct, having individual time constants. The curves corresponding to initial conditions falling on an eigenvector (a principal axis of the elliptical contours) are pure exponentials, whereas the curve corresponding to the initial condition between the eigenvectors is a sum of two exponentials.

Which algorithm converges faster? For some initial conditions, LMS converges faster than LMS/Newton. For other initial conditions, LMS is slower than LMS/Newton. Yet for other initial conditions, LMS has multiple modes, some faster than LMS/Newton and some slower than it. The question is: which is faster, LMS or LMS/Newton?

7. Excess error energy

The question 'which is faster' prompts a new look at the learning curve and at the issue of learning time. Regarding the learning curves of Fig. 7, it is clear that during the transient, the MSE is excessive. One strives to reduce this MSE to ξ_{min} as fast as possible. The area under the curve and above ξ_{min}, the shaded area, is defined as the excess error energy. This area is a sum of MSE over time. One would like this area to be as small as possible.

The learning curve shown in Fig. 7(a) is an exponential having a single time constant. The area under this curve, the excess error energy, is equal to the amplitude of the exponential (its initial excess MSE) multiplied by the time constant. The learning time for this curve may be considered to be four time constants. This is somewhat arbitrary, but reasonable. Accordingly, the learning time for the single

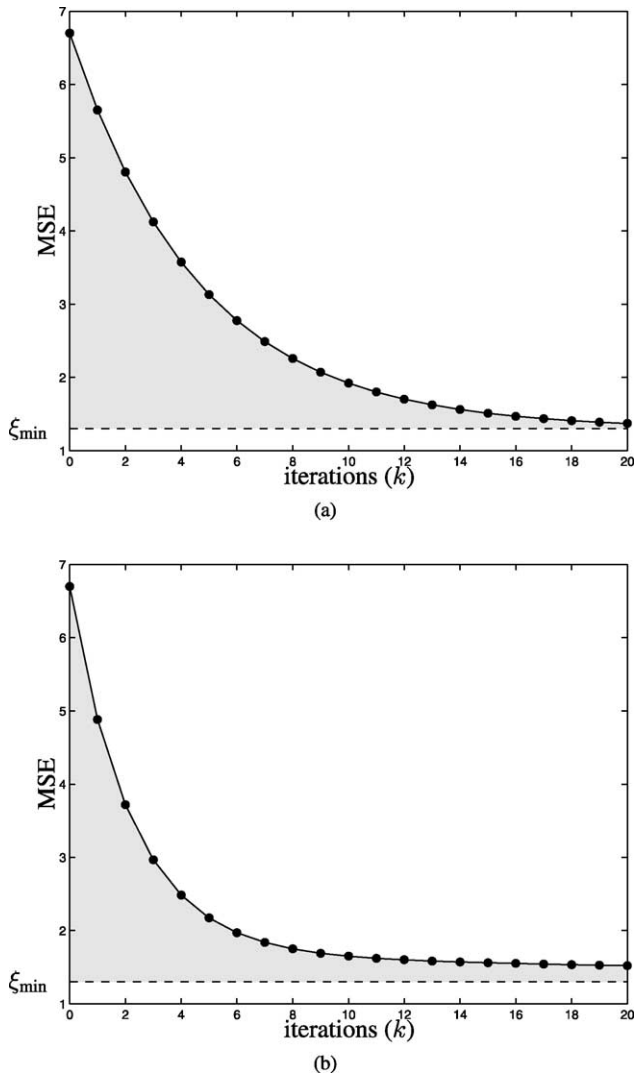


Fig. 7. Idealized learning curves (no gradient noise): (a) single-exponential, (b) two-exponential. The shaded area represents the excess error energy.

exponential is defined as

$$(\text{learning time}) \triangleq 4 \times \frac{(\text{excess error energy})}{(\text{initial excess MSE})}. \tag{26}$$

The learning curve shown in Fig. 7(b) is a sum of two exponentials. Its excess error energy is the same as that of the single exponential learning curve of Fig. 7(a), and the initial excess MSE of both curves is the same. The learning times of both curves are therefore the same and are both given by Eq. (26). In general, the learning time of a learning curve which is a sum of any number of exponentials will be defined hereby as given by Eq. (26). Starting from the same initial MSE, the convergence times of two different learning curves are defined as being identical if their respective excess error energies are equal. Excess error energy is proportional to learning time as it is defined here.

The question remains, which has the greater learning time, LMS or LMS/Newton? Referring to Fig. 6, one can see that LMS/Newton has a fixed learning time, but the learning time of LMS depends on initial conditions. Sometimes LMS is faster than LMS/Newton, sometimes slower. Under worst-case initial conditions, LMS can have a much greater learning time than LMS/Newton. However, it has been shown (Widrow & Kamenetsky, 2003) that with random initial conditions, the average learning time with LMS is identical to the unique learning time of LMS/Newton. LMS/Newton is optimal and its performance is the benchmark. It is important to realize that the average performance of LMS is identical to that of LMS/Newton, when comparing both algorithms starting from the same randomly chosen initial conditions. One must be careful to note that this does *not* mean that the learning curves of LMS and LMS/Newton are identical, even when averaged over initial conditions. On the contrary, it can be shown that the average initial convergence of LMS is faster than that of LMS/Newton, whereas the average final convergence of LMS is slower than that of LMS/Newton (Widrow & Kamenetsky, 2003). However, their average learning times and average excess error energies are the same.

8. LMS and LMS/Newton in a nonstationary environment

With statistically stationary inputs, the quadratic performance surface is fixed, and the Wiener solution is fixed. With nonstationary inputs, this surface changes randomly, and the Wiener solution is not fixed but is a randomly moving target. There is an analogy between, on the one hand, transient adaptation from an ensemble of random initial conditions toward a fixed Wiener target and, on the other hand, steady state adaptation toward a randomly moving Wiener target. In this section, a comparison is made of the performances of the LMS algorithm and the LMS/Newton algorithm when adapting with nonstationary inputs of a simple form.

Filtering nonstationary signals is a major area of application for adaptive systems. When the statistical character of an input signal changes gradually, randomly, and unpredictably, a filtering system that can automatically optimize its input-output response in accord with the requirements of the input signal could yield superior performance relative to that of a fixed, non-adaptive system. The performance of the conventional steepest descent LMS algorithm is compared here with LMS/Newton (which, as demonstrated above, possesses optimality qualities), when both algorithms are used to adapt transversal filters with nonstationary inputs. The nonstationary situations to be studied are highly simplified, but they retain the essence of the problem that is common to more complicated and realistic situations.

The example considered here involves modeling or identifying an unknown time-varying system by an adaptive LMS transversal filter of length n. The unknown system is

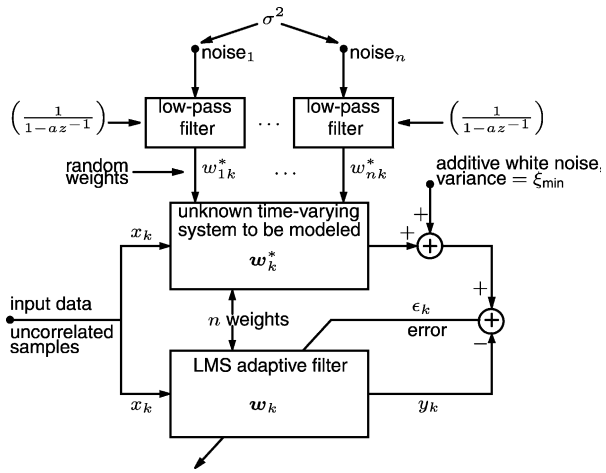


Fig. 8. Modeling an unknown time-varying system.

assumed to be a transversal filter of the same length n whose weights (impulse response values) vary as independent stationary ergodic first-order Markov processes, as indicated in Fig. 8. The input signal x_k is assumed to be stationary and ergodic. Additive output noise, assumed to be stationary and ergodic, of mean zero and of variance $\xi_{\min}$, prevents a perfect match between the unknown system and the adaptive system. The minimum MSE is, therefore, $\xi_{\min}$, and it is achieved whenever the weights of the adaptive filter, $\boldsymbol{w}_k$, match those of the unknown system. The latter are at every instant the optimal values for the corresponding weights of the adaptive filter and are designated as $\boldsymbol{w}_k^*$, the time index indicating that the unknown 'target' to be tracked is time-varying.

The components of $\boldsymbol{w}_k^*$ are generated by passing independent white noises of variance σ^2 through identical one-pole low-pass filters. The components of $\boldsymbol{w}_k^*$ therefore vary as independent first-order Markov processes. The formation of $\boldsymbol{w}_k^*$ is illustrated in Figs. 8 and 9.

According to the scheme of Fig. 8, minimizing the MSE causes the adaptive weight vector $\boldsymbol{w}_k$ to attempt to best match the unknown $\boldsymbol{w}_k^*$ on a continual basis. The $\mathbf{R}$ matrix, dependent only on the statistics of $\boldsymbol{x}_k$, is constant even as $\boldsymbol{w}_k^*$ varies.

The desired response of the adaptive filter, d_k, is nonstationary, being the output of a time-varying system. The minimum MSE, $\xi_{\min}$, is constant. Thus the MSE function, a quadratic bowl, varies in position while its eigenvalues, eigenvectors, and $\xi_{\min}$ remain constant.

In order to study this form of nonstationary adaptation both analytically and by computer simulation, a model comprising an ensemble of nonstationary adaptive processes has been defined and constructed as illustrated in Fig. 9. Throughout the ensemble, the unknown filters to be modeled are all identical and have the same time-varying weight vector $\boldsymbol{w}_k^*$. Each ensemble member has its own independent input signal going to both the unknown system

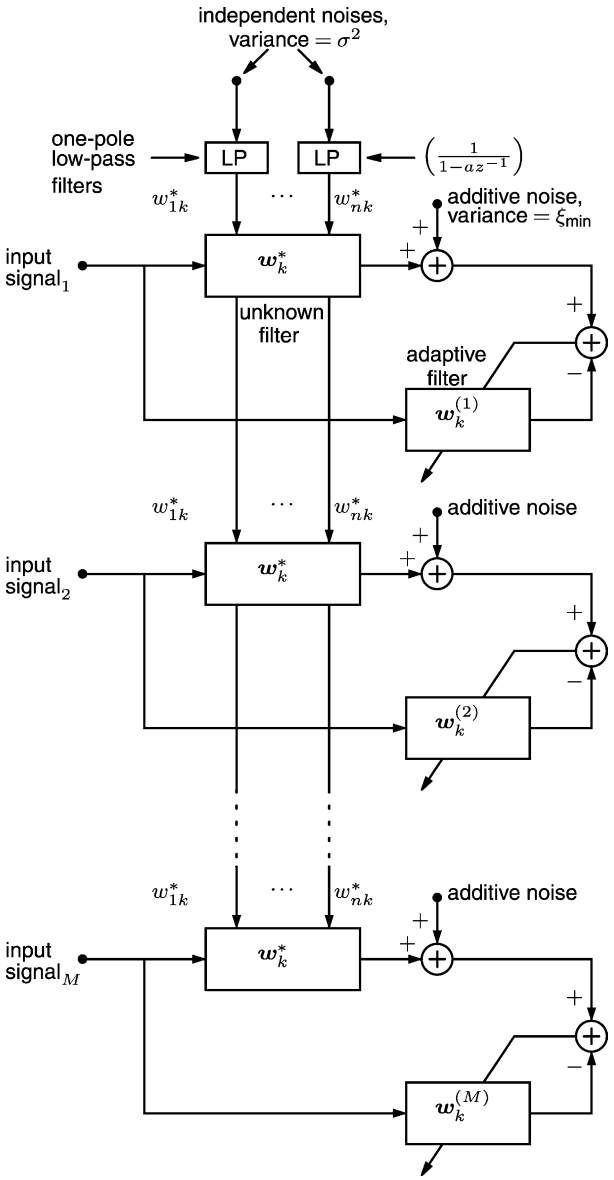


Fig. 9. An ensemble of nonstationary adaptive processes.

and the corresponding adaptive system. The effect of output noise in the unknown systems is obtained by the addition of independent noises of variance $\xi_{\min}$. All of the adaptive filters are assumed to start with the same initial weight vector $\boldsymbol{w}_0$; each develops its own weight vector over time in attempting to pursue the moving Markovian target $\boldsymbol{w}_k^*$.

For a given adaptive filter, the weight-vector tracking error at the kth instant is $\boldsymbol{v}_k \triangleq \boldsymbol{w}_k - \boldsymbol{w}_k^*$. This error is due to both the effects of gradient noise and weight-vector lag and may be expressed as

$$\boldsymbol{v}_k = \boldsymbol{w}_k - \boldsymbol{w}_k^* = \underbrace{(w_k - E[w_k])}_{\text{weight-vector noise}} + \underbrace{(E[w_k] - w_k^*)}_{\text{weight-vector lag}} . \qquad (27)$$

The expectations are averages over the ensemble. Eq. (27) identifies the two components of the error. Any difference

between the ensemble mean of the adaptive weight vectors and the target value $\boldsymbol{w}_k^*$ is due to lag in the adaptive process, while the deviation of the individual adaptive weight vectors about the ensemble mean is due to gradient noise.

Weight-vector error causes an excess MSE. The ensemble average excess MSE at the kth instant is

$$(\text{average excess MSE})_k = E[(\boldsymbol{w}_k - \boldsymbol{w}_k^*)^{\mathrm{T}}\mathbf{R}(\boldsymbol{w}_k - \boldsymbol{w}_k^*)]. \tag{28}$$

Using Eq. (27), this can be expanded as follows:

$$\begin{aligned}&(\text{average excess MSE})_k\\ &\quad = E[(\boldsymbol{w}_k - E[\boldsymbol{w}_k])^{\mathrm{T}}\mathbf{R}(\boldsymbol{w}_k - E[\boldsymbol{w}_k])] + E[(E[\boldsymbol{w}_k])\\ &\qquad - \boldsymbol{w}_k^*)^{\mathrm{T}}\mathbf{R}(E[\boldsymbol{w}_k] - \boldsymbol{w}_k^*)] + 2E[(\boldsymbol{w}_k - E[\boldsymbol{w}_k])^{\mathrm{T}}\\ &\qquad \times R(E[\boldsymbol{w}_k] - \boldsymbol{w}_k^*)].\end{aligned} \tag{29}$$

Expanding the last term of Eq. (29) and simplifying since $\boldsymbol{w}_k^*$ is constant over the ensemble,

$$\begin{aligned}&2E[\boldsymbol{w}_k^{\mathrm{T}}\mathbf{R}E[\boldsymbol{w}_k] - \boldsymbol{w}_k^{\mathrm{T}}\mathbf{R}w_k^* - E[\boldsymbol{w}_k]^{\mathrm{T}}\mathbf{R}E[\boldsymbol{w}_k] + E[\boldsymbol{w}_k]^{\mathrm{T}}\mathbf{R}w_k^*]\\ &\quad = 2(E[\boldsymbol{w}_k]^{\mathrm{T}}\mathbf{R}E[\boldsymbol{w}_k] - E[\boldsymbol{w}_k]^{\mathrm{T}}\mathbf{R}E[\boldsymbol{w}_k] - E[\boldsymbol{w}_k]^{\mathrm{T}}\\ &\qquad \times \mathbf{R}\boldsymbol{w}_k^* + E[\boldsymbol{w}_k]^{\mathrm{T}}\mathbf{R}\boldsymbol{w}_k^*) = 0.\end{aligned} \tag{30}$$

Therefore, Eq. (29) becomes

$$\begin{aligned}&(\text{average excess MSE})_k\\ &\quad = E[(\boldsymbol{w}_k^* - E[\boldsymbol{w}_k])^{\mathrm{T}}\mathbf{R}(\boldsymbol{w}_k - E[\boldsymbol{w}_k])] + E[(E[\boldsymbol{w}_k]\\ &\qquad - \boldsymbol{w}_k^*)^{\mathrm{T}}\mathbf{R}(E[\boldsymbol{w}_k] - \boldsymbol{w}_k^*)].\end{aligned} \tag{31}$$

The average excess MSE is thus a sum of components due to both gradient noise and lag:

$$\begin{aligned}&(\text{average excess MSE due to lag})_k\\ &\quad = E[(E[\boldsymbol{w}_k] - \boldsymbol{w}_k^*)^{\mathrm{T}}\mathbf{R}(E[\boldsymbol{w}_k] - \boldsymbol{w}_k^*)]\\ &\quad = E[(E[\boldsymbol{w}'_k] - \boldsymbol{w}'{*}_k)^{\mathrm{T}} \wedge (E[\boldsymbol{w}'_k] - \boldsymbol{w}'{*}_k)]\end{aligned} \tag{32}$$

$$\begin{aligned}&(\text{average excess MSE due to gradient noise})_k\\ &\quad = E[(\boldsymbol{w}_k - E[\boldsymbol{w}_k]^{\mathrm{T}})\mathbf{R}(\boldsymbol{w}_k - E[\boldsymbol{w}_k])]\\ &\quad = E[(\boldsymbol{w}'_k - E[w'_k])^{\mathrm{T}} \wedge (\boldsymbol{w}'_k - E[\boldsymbol{w}'_k])],\end{aligned} \tag{33}$$

where $\boldsymbol{w}'_k \triangleq \mathbf{Q}^{\mathrm{T}}\boldsymbol{w}_k$, $w'{*}_k \triangleq \mathbf{Q}^{\mathrm{T}}\boldsymbol{w}_k^*$, $\wedge$ is a diagonal matrix of the eigenvalues of $\mathbf{R}$, and $\mathbf{Q}$ is the modal matrix of $\mathbf{R}$. The total misadjustment is therefore a sum of two components, that due to lag and that due to gradient noise. These components of misadjustment have been evaluated by Widrow et al. (1976). The total misadjustment for adaptation with the LMS algorithm is

$$\begin{aligned}M_{\text{sum}} &= (\text{misadjustment due to gradient noise})\\ &\quad + (\text{misadjustment due to lag})\\ &= \mu\mathrm{Tr}(\boldsymbol{R}) + \frac{n\sigma^2}{4\mu\xi_{\min}}.\end{aligned} \tag{34}$$

Since M_{sum} is convex in μ, an optimal choice of μ that minimizes M_{sum} can be obtained by differentiating M_{sum} with respect to μ and setting the derivative to zero. Optimization takes place when the two terms of Eq. (34) are made equal. When this happens, the loss in performance from adapting too rapidly (due to gradient noise) is equal to the loss in performance from adapting too slowly (due to lag).

It is interesting to note that M_{sum} in Eq. (34) depends on the choice of the parameter μ and on the statistical properties of the nonstationary environment but does not depend on the spread of the eigenvalues of the $\boldsymbol{R}$ matrix. It is no surprise, therefore, that when the components of misadjustment are evaluated for the LMS/Newton algorithm operating in the very same environment, the expression for M_{sum} for LMS/Newton turns out to be

$$\begin{aligned}M_{\text{sum}} &= (\text{misadjustment due to gradient noise})\\ &\quad + (\text{misadjustment due to lag})\\ &= \mu\mathrm{Tr}(\boldsymbol{R}) + \frac{n\sigma^2}{4\mu\xi_{\min}},\end{aligned} \tag{35}$$

which is the same as Eq. (34). From this we may conclude that the performance of the LMS algorithm is equivalent to that of the LMS/Newton algorithm when both are operating with the same choice of μ in the same nonstationary environment, wherein they are tracking a first-order Markov target. The optimum value of μ is the same for LMS/Newton as for LMS. Since LMS/Newton is optimal, we may conclude that the LMS algorithm is also optimal when operating in a first-order Markov nonstationary environment. And LMS is likely to be optimal or close to it when operating in many other types of nonstationary environments, although this has not yet been proven.

9. Conclusion

An adaptive algorithm is like an engine whose fuel is input data. Two algorithms adapting the same number of weights and operating with the same misadjustment can be compared in terms of their consumption of data. The more efficient algorithm consumes less data, i.e. converges faster. On this basis, the LMS/Newton algorithm has the highest statistical efficiency that can be obtained. The LMS/Newton algorithm therefore can serve as a benchmark for statistical efficiency against which all other algorithms can be compared.

The role played by LMS/Newton in adaptive systems is analogous to that played by the Carnot engine in thermodynamics. They both do not exist physically. But

their performances limit the performances of all practical systems, adaptive and thermodynamic, respectively.

The LMS/Newton algorithm uses learning data most efficiently. No other least squares learning algorithm can be more efficient. The LMS algorithm performs identically to LMS/Newton when all eigenvalues of the input autocorrelation matrix (the $\boldsymbol{R}$ matrix) are equal. In most practical cases, however, the eigenvalues are not equal. Regardless of eigenvalue spread, the LMS algorithm performs equivalently, on average, to LMS/Newton in nonstationary environments and under transient learning conditions with random initial conditions. However, under 'worst case' initial conditions, LMS can converge much more slowly than LMS/Newton. Under 'best case' initial conditions, LMS converges much faster than LMS/Newton. On average, their convergence rates are equivalent in terms of their excess error energies. Along with its simplicity, ease of implementation, and robustness, the equivalent performance between LMS and LMS/Newton is one of the major reasons for the popularity of the LMS algorithm. Dr. Widrow has discussed this with Dr Paul Werbos, and both conclude that the backpropagation algorithm is popular for similar reasons.

References

Butterweck, H. J. (2001). A wave theory of long adaptive filters. *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications*, *48*(6), 739–747.

Dabeer, O., & Masry, E. (2002). Analysis of mean-square error and transient speed of the LMS adaptive algorithm. *IEEE Transactions on Information Theory*, *48*(7), 1873–1894.

Ghogho, M., Ibnkahla, M., & Bershad, N. J. (1998). Analytic behavior of the LMS adaptive line enhancer for sinusoids corrupted by multiplicative and additive noise. *IEEE Transactions on Signal Processing*, *46*(9), 2386–2393.

Kailath, T., Sayed, A. H., & Hassibi, B. (2000). *Linear estimation*. Upper Saddle River, NJ: Prentice Hall.

Proakis, J. G. (2001). *Digital communications* (4th ed). New York: McGraw-Hill, Chapter 11.

Werbos, P. J (1974). *Beyond regression, new tools for prediction and analysis in the behavioral sciences*. PhD thesis, Harvard University, Cambridge, MA.

Widrow, B., Glover, J. R., Jr., McCool, J. M., Kaunitz, J., Williams, C. S., Hearn, R. H., Zeidler, J. R., Dong, E., Jr., & Goodlin, R. C. (1975). Adaptive noise cancelling. Principles and applications. *Proceedings of the IEEE*, *63*(12), 1692–1716.

Widrow, B., & Hoff, M. E. (1960). Adaptive switching circuits. *IRE WESCON Convention Record*, *4*, 96–104.

Widrow, B., & Kamenetsky, M. (2003). On the efficiency of adaptive algorithms. In S. Haykin, & B. Widrow (Eds.), *Least-mean-square adaptive filters*. New York: Wiley.

Widrow, B., Mantey, P., Griffiths, L., & Goode, B. (1967). Adaptive antenna systems. *Proceedings of the IEEE*, *55*(12), 2143–2159.

Widrow, B., McCool, J., Larimore, M. G., & Johnson, C. R., Jr. (1976). Stationary and nonstationary learning characteristics of the LMS adaptive filter. *Proceedings of the IEEE*, *64*(8), 1151–1162.

Widrow, B., & Stearns, S. D. (1985). *Adaptive signal processing*. Upper Saddle River, NJ: Prentice Hall.

Widrow, B., & Walach, E. (1984). On the statistical efficiency of the LMS algorithm with nonstationary inputs. *IEEE Transactions on Information Theory*, *IT-30*(2), 211–221.

Widrow, B., & Walach, E. (1996). *Adaptive inverse control*. Upper Saddle River, NJ: Prentice Hall.

Yousef, N. R., & Sayed, A. H. (2001). A unified approach to the steady-state and tracking analyses of adaptive filters. *IEEE Transactions on Signal Processing*, *49*(2), 314–324.