CHAPTER 10

LMS Algorithm

In Chapters 10–14 we start to develop the theory of adaptive algorithms by studying *stochastic-gradient* methods. These methods are obtained from steepest-descent implementations by replacing the required gradient vectors and Hessian matrices by some suitable approximations. Different approximations lead to different algorithms with varied degrees of complexity and performance properties. The resulting methods will be generically called *stochastic-gradient algorithms* since, by employing estimates for the gradient vector, the update directions become subject to random fluctuations that are often referred to as *gradient noise*.

Stochastic-gradient algorithms serve at least two purposes. First, they avoid the need to know the *exact signal statistics* (e.g., covariances and cross-covariances), which are necessary for a successful steepest-descent implementation but are nevertheless rarely available in practice. Stochastic-gradient methods achieve this feature by means of a *learning mechanism* that enables them to estimate the required signal statistics. Second, these methods possess a *tracking mechanism* that enables them to track variations in the signal statistics. The two combined capabilities of learning and tracking are the main reasons behind the widespread use of stochastic-gradient methods (and the corresponding adaptive filters). It is because of these abilities that we tend to describe adaptive filters as "smart systems"; smart in the sense that they can learn the statistics of the underlying signals and adjust their behavior to variations in the "environment" in order to keep the performance level at check.

In the body of this chapter and the subsequent chapters, we describe a handful of stochastic-gradient algorithms, including the least-mean-squares (LMS) algorithm, the normalized least-mean-squares (NLMS) algorithm, the affine projection algorithm (APA), and the recursive least-squares algorithm (RLS). In Probs. III.26–III.38 we devise several other stochastic-gradient methods. By the end of the presentation, the reader will have had sufficient exposition to the procedure that is commonly used to motivate and derive adaptive filters.

10.1 MOTIVATION

We start our discussions by reviewing the linear estimation problem of Sec. 8.1 and the corresponding steepest-descent methods of Chapter 8. Thus, let $\boldsymbol{d}$ be a zero-mean scalar-valued random variable with variance $\sigma_d^2 = \mathsf{E}\,|\boldsymbol{d}|^2$. Let further $\boldsymbol{u}^*$ be a zero-mean $M \times 1$ random variable with a positive-definite covariance matrix, $R_u = \mathsf{E}\,\boldsymbol{u}^*\boldsymbol{u} > 0$. The $M \times 1$ cross-covariance vector of $\boldsymbol{d}$ and $\boldsymbol{u}$ is denoted by $R_{du} = \mathsf{E}\,\boldsymbol{d}\boldsymbol{u}^*$. We know from Sec. 8.1 that the weight vector w that solves

$$\min_{w} \mathsf{E}\,|\boldsymbol{d} - \boldsymbol{u}w|^2 \tag{10.1}$$

Adaptive Filters, by Ali H. Sayed
Copyright © 2008 John Wiley & Sons, Inc.

is given by

$$\boxed{w^o = R_u^{-1} R_{du}} \tag{10.2}$$

and that the resulting minimum mean-square error is

$$\boxed{\text{m.m.s.e.} \;=\; \sigma_d^2 \;-\; R_{ud} R_u^{-1} R_{du}} \tag{10.3}$$

In Chapter 8 we developed several steepest-descent methods that approximate w^o iteratively, until eventually converging to it. For example, in Sec. 8.1 we studied the following recursion with a constant step-size,

$$\boxed{w_i = w_{i-1} + \mu[R_{du} - R_u w_{i-1}], \quad w_{-1} = \text{initial guess}} \tag{10.4}$$

where the update direction, $R_{du} - R_u w_{i-1}$, was seen to be equal to the negative conjugate-transpose of the gradient vector of the cost function at w_{i-1}, i.e.,

$$R_{du} - R_u w_{i-1} = -[\nabla_w J(w_{i-1})]^* \tag{10.5}$$

where

$$J(w) \;\triangleq\; \mathsf{E}\,|\boldsymbol{d} - \boldsymbol{u}w|^2$$

In Sec. 9.7 we allowed for an iteration-dependent step-size, $\mu(i)$, and studied the recursion

$$\boxed{w_i = w_{i-1} + \mu(i)[R_{du} - R_u w_{i-1}], \quad w_{-1} = \text{initial guess}} \tag{10.6}$$

and in Sec. 9.8 we studied Newton's recursion,

$$\boxed{w_i = w_{i-1} + \mu R_u^{-1}[R_{du} - R_u w_{i-1}], \quad w_{-1} = \text{initial guess}} \tag{10.7}$$

where R_u^{-1} resulted from using the inverse of the Hessian matrix of $J(w)$, namely,

$$R_u = \nabla_w^2 \, J(w_{i-1}) \;=\; \nabla_{w^*} \, [\nabla_w J(w_{i-1})]$$

More generally, when regularization is employed and when the step-size is also allowed to be iteration-dependent, the recursion for Newton's method is replaced by

$$\boxed{w_i = w_{i-1} + \mu(i) \, [\epsilon(i)\mathrm{I} \;+\; R_u]^{-1} \, [R_{du} - R_u w_{i-1}]} \tag{10.8}$$

for some positive scalars $\{\epsilon(i)\}$; they could be set to a constant value for all i, say, $\epsilon(i) = \epsilon$.

Now all the steepest-descent formulations described above, i.e., (10.4), (10.6), (10.7) and (10.8), rely explicitly on knowledge of $\{R_{du}, R_u\}$. This fact constitutes a limitation in practice and serves as a motivation for the development of stochastic-gradient algorithms for two reasons:

1. **Lack of statistical information**. First, the quantities $\{R_{du}, R_u\}$ are rarely available in practice. As a result, the true gradient vector, $\nabla_w J(w_{i-1})$, and the true Hessian matrix, $\nabla_w^2 J(w_{i-1})$, cannot be evaluated exactly and a true steepest-descent implementation becomes impossible. Stochastic-gradient algorithms replace the gradient vector and the Hessian matrix by approximations for them. There are several ways for obtaining such approximations. The better the approximation, the closer we expect the performance of the resulting stochastic-gradient algorithm to be to that of the original steepest-descent method. In Parts IV (*Mean-Square Performance*) and V

(*Transient Performance*) we shall study and quantify the degradation in performance that occurs as a result of such approximations.

2. **Variation in the statistical information.** Second, and even more important, the quantities $\{R_{du}, R_u, \}$ tend to vary with time.[4] In this way, the optimal weight vector w^o will also vary with time. It turns out that stochastic-gradient algorithms provide a mechanism for tracking such variations in the signal statistics.

We therefore move on to motivate and develop several stochastic-gradient algorithms.

10.2 INSTANTANEOUS APPROXIMATION

Assume that we have access to several observations of the random variables $\boldsymbol{d}$ and $\boldsymbol{u}$ in (10.1), say, $\{d(0), d(1), d(2), d(3), \ldots\}$ and $\{u_0, u_1, u_2, u_3, \ldots\}$. We refer to the $\{u_i\}$ as regressors. Observe that, in conformity with our notation in this book, we are using the boldface letter $\boldsymbol{d}$ to refer to the random variable, and the normal letter d to refer to observations (or realizations) of it. Likewise, we write $\boldsymbol{u}$ for the random vector and u for observations of it.

One of the simplest approximations for $\{R_{du}, R_u\}$ is to use the instantaneous values

$$\widehat{R}_{du} \;=\; d(i)u_i^*, \qquad \widehat{R}_u \;=\; u_i^* u_i \tag{10.9}$$

This construction simply amounts to dropping the expectation operator from the actual definitions, $R_{du} = \mathsf{E}\,\boldsymbol{d}\boldsymbol{u}^*$ and $R_u = \mathsf{E}\,\boldsymbol{u}^*\boldsymbol{u}$, and replacing the random variables $\{\boldsymbol{d}, \boldsymbol{u}\}$ by the observations $\{d(i), u_i\}$ at iteration i. In this way, the gradient vector (10.5) is approximated by the instantaneous value

$$-\left[\nabla_w J(w_{i-1})\right]^* \;\approx\; d(i)u_i^* - u_i^* u_i w_{i-1} = u_i^*[d(i) - u_i w_{i-1}]$$

and the corresponding steepest-descent recursion (10.4) becomes

$$w_i = w_{i-1} + \mu u_i^*[d(i) - u_i w_{i-1}]\,, \quad w_{-1} = \text{initial guess} \tag{10.10}$$

We continue to write w_i to denote the estimate that is obtained via this approximation procedure although, of course, w_i in (10.10) is different from the w_i that is obtained from the steepest-descent algorithm (10.4): the former is based on using instantaneous approximations whereas the latter is based on using $\{R_{du}, R_u\}$. We do so in order to avoid an explosion of notation; the distinction between both estimates is usually clear from the context.

The stochastic-gradient approximation (10.10) is one of the most widely used adaptive algorithms in current practice due to its striking simplicity. It is widely known as the least-mean-squares (LMS) algorithm, or sometimes as the Widrow-Hoff algorithm in honor of its originators.

[4] We have so far interpreted the index i in a steepest-descent recursion, e.g., as in (10.4), as merely an iteration index. In the adaptive context, however, it will become common to interpret i as a time index in which case w_i is interpreted as the weight estimate at time i.

Algorithm 10.1 (LMS Algorithm) Consider a zero-mean random variable $\boldsymbol{d}$ with realizations $\{d(0), d(1), \ldots\}$, and a zero-mean random row vector $\boldsymbol{u}$ with realizations $\{u_0, u_1, \ldots\}$. The optimal weight vector w^o that solves

$$\min_{w} \mathsf{E}\,|\boldsymbol{d} - \boldsymbol{u}w|^2$$

can be approximated iteratively via the recursion

$$w_i = w_{i-1} + \mu u_i^*[d(i) - u_i w_{i-1}], \quad i \geq 0, \quad w_{-1} = \text{initial guess}$$

where μ is a positive step-size (usually small).

10.3 COMPUTATIONAL COST

A useful property of LMS is its computational simplicity. The evaluations that follow for the number of computations that are required by LMS are intended to provide an approximate idea of its computational complexity. While there can be different ways to perform specific calculations, the resulting overall filter complexity will be of the same order of magnitude (and often quite similar) to the values we derive in this section for LMS, and in subsequent sections for other adaptive filters.

Note that each step of the LMS algorithm requires a handful of straightforward computations, which are explained below:

1. Each iteration (10.10) requires the evaluation of the inner product $u_i w_{i-1}$, between two vectors of size M each. This calculation requires M complex multiplications and $(M-1)$ complex additions. Using the fact that one complex multiplication requires four real multiplications and two real additions, while one complex addition requires two real additions, we find that the evaluation of this inner product requires $4M$ real multiplications and $4M-2$ real additions.

2. The algorithm also requires the evaluation of the scalar $d(i) - u_i w_{i-1}$. This calculation amounts to one complex addition, i.e., 2 real additions.

3. Evaluation of the product $\mu[d(i) - u_i w_{i-1}]$, where μ is a real scalar, requires two real multiplications when the data is complex-valued. Usually, μ is chosen as a power of 2^{-1}, say, 2^{-m} for some integer $m > 0$. In this case, multiplying μ by $[d(i) - u_i w_{i-1}]$ can be implemented digitally very efficiently by means of shift registers, and we could therefore ignore the cost of this multiplication. In the text, we choose to account for the case of arbitrary step-sizes.

4. The algorithm further requires multiplying the scalar $\mu[d(i) - u_i w_{i-1}]$ by the vector u_i^*. This requires M complex multiplications and, therefore, $4M$ real multiplications and $2M$ real additions.

5. Finally, the addition of the two vectors w_{i-1} and $\mu u_i^*[d(i) - u_i w_{i-1}]$ requires M complex additions, i.e., $2M$ real additions.

In summary, for general complex-valued signals, LMS requires $8M+2$ real multiplications and $8M$ real additions per iteration. On the other hand, for real-valued data, LMS requires $2M+1$ real multiplications and $2M$ real additions per iteration.

10.4 LEAST-PERTURBATION PROPERTY

The LMS algorithm was derived in Sec. 10.2 as an approximate iterative solution to the linear least-mean-squares estimation problem (10.1), in the sense that it was obtained by replacing the actual gradient vector in the steepest-descent implementation (10.4) by an instantaneous approximation for it. It turns out that LMS could have been derived in a different manner as the *exact* (not approximate) solution of a well-defined estimation problem; albeit one with a different cost criterion. We shall pursue this derivation later in Chapter 45 — see Sec. 45.4, when we study robust adaptive filters. Until then, we shall continue to treat LMS as a stochastic-gradient algorithm and proceed to highlight some of its properties.

One particular property is that LMS can also be regarded as the exact solution to a *local* (as opposed to a global) optimization problem. To see this, assume that we have available some weight estimate at time $i-1$, say, w_{i-1}, and let $\{d(i), u_i\}$ denote the newly measured data. Let w_i denote a new weight estimate that we wish to compute from the available data $\{d(i), u_i, w_{i-1}\}$ in the following manner.

Let μ denote a given positive number and define two estimation errors: the *a priori output* estimation error

$$\boxed{e(i) \triangleq d(i) - u_i w_{i-1}} \tag{10.11}$$

and the *a posteriori output* estimation error

$$\boxed{r(i) \triangleq d(i) - u_i w_i} \tag{10.12}$$

The former measures the error in estimating $d(i)$ by using $u_i w_{i-1}$, i.e., by using the available weight estimate prior to the update, while the latter measures the error in estimating $d(i)$ by using $u_i w_i$, i.e., by using the new weight estimate. We then seek a w_i that solves the constrained optimization problem:

$$\boxed{\min_{w_i} \ \|w_i - w_{i-1}\|^2 \quad \text{subject to} \quad r(i) = (1 - \mu\|u_i\|^2)e(i)} \tag{10.13}$$

In other words, we seek a solution w_i that is closest to w_{i-1} in the Euclidean norm sense and subject to an equality constraint between $r(i)$ and $e(i)$. The constraint is most relevant when the step-size μ is small enough to satisfy $\left|1 - \mu\|u_i\|^2\right| < 1$, i.e., when

$$0 < \mu\|u_i\|^2 < 2 \qquad \text{for all } i \tag{10.14}$$

This is because, when (10.14) holds, the magnitude of the *a posteriori* error, $r(i)$, will always be less than that of the *a priori* error, $e(i)$, i.e., $|r(i)| < |e(i)|$ or, equivalently, $u_i w_i$ will be a better estimate for $d(i)$ than $u_i w_{i-1}$ (except when $e(i) = 0$, in which case $r(i) = 0$ also). One could of course impose different kinds of constraints on $r(i)$ than (10.13); these will not lead to the LMS algorithm but to other algorithms.

The solution of (10.13) can be obtained from first principles as follows. Let $\delta w = (w_i - w_{i-1})$. Then

$$\begin{aligned} u_i \delta w &= u_i w_i - u_i w_{i-1} \\ &= [u_i w_i - d(i)] + [d(i) - u_i w_{i-1}] \\ &= -r(i) + e(i) \\ &= \mu\|u_i\|^2 e(i) \end{aligned}$$

where in the last equality we used the constraint (10.13). This calculation shows that the optimization problem (10.13) is equivalent to determining a vector δw of smallest Euclidean norm that satisfies $u_i \delta w = \mu \|u_i\|^2 e(i)$, i.e.,

$$\min_{\delta w} \ \|\delta w\|^2 \quad \text{subject to} \quad u_i \delta w = \mu \|u_i\|^2 e(i) \tag{10.15}$$

The constraint $u_i \delta w = \mu \|u_i\|^2 e(i)$ amounts to an under-determined linear system of equations in δw, and it therefore admits infinitely many solutions. Among all solutions δw we desire to determine one that has the smallest Euclidean norm. To do so, we distinguish between two cases:

1. $\|u_i\|^2 = 0$. In this case, $u_i = 0$ and the constraint in (10.15) is satisfied by any δw. The δw with smallest Euclidean norm is then $\delta w = 0$, so that $w_i = w_{i-1}$.

2. $\|u_i\|^2 \neq 0$. In this case, it is easy to find at least one solution to $u_i \delta w = \mu \|u_i\|^2 e(i)$. Indeed,

$$\delta w^o = \mu u_i^* e(i) \tag{10.16}$$

is one such solution since if we multiply δw^o by u_i from the left we get

$$u_i \delta w^o = \mu \|u_i\|^2 e(i)$$

However, there are infinitely many other solutions to this equation. To see this, let $\delta w^o + z$ denote any other possible solution, say,

$$u_i(\delta w^o + z) = \mu \|u_i\|^2 e(i)$$

Now since δw^o is a solution, we find that z must satisfy $u_i z = 0$. This means that all solutions to $u_i \delta w = \mu \|u_i\|^2 e(i)$ can be obtained from δw^o by adding to it any column vector z that is orthogonal to u_i (and there are infinitely many such vectors). Still, among all solutions $\{\delta w^o + z\}$, it can be verified that the δw^o in (10.16) has the smallest Euclidean norm since

$$\begin{aligned} \|\delta w^o + z\|^2 &= \|\delta w^o\|^2 + \|z\|^2 + (\delta w^o)^* z + z^* \delta w^o \\ &= \|\delta w^o\|^2 + \|z\|^2 + \mu (u_i z) e^*(i) + \mu (u_i z)^* e(i) \\ &= \|\delta w^o\|^2 + \|z\|^2 + 0 + 0 \\ &> \|\delta w^o\|^2 \end{aligned}$$

Finally, using $\delta w^o = w_i - w_{i-1}$ and (10.16) we arrive at the following expression for w_i:

$$w_i = w_{i-1} + \mu u_i^* [d(i) - u_i w_{i-1}]$$

which coincides with the LMS recursion (10.10). We have therefore established that LMS is the solution to the localized optimization problem (10.13).

10.5 APPLICATION: ADAPTIVE CHANNEL ESTIMATION

Before proceeding to the derivation of other similar stochastic-gradient algorithms, we find it instructive to describe how such algorithms are useful in the context of several applications, including channel estimation, linear equalization and decision-feedback equalization. These are the same applications that we studied before in Secs. 5.2, 5.4, 6.3 and 6.4, except that now we are going to solve them in an iterative (i.e., adaptive) manner. We start with the problem of channel estimation.

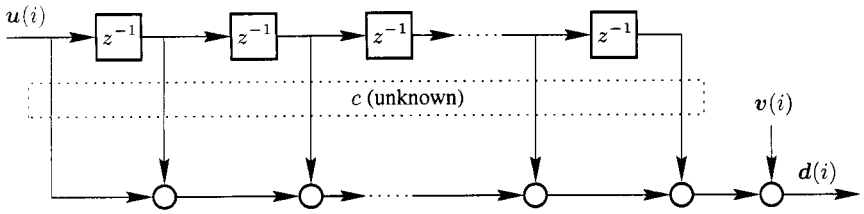

FIGURE 10.1 Noisy measurements of an FIR channel with an impulse response vector c.

Figure 10.1 shows an FIR channel excited by a zero-mean random sequence $\{\boldsymbol{u}(i)\}$. Its output is another zero-mean random sequence $\{\boldsymbol{d}(i)\}$. At any time instant i, the state of the channel is captured by the regressor

$$\boldsymbol{u}_i = \begin{bmatrix} \boldsymbol{u}(i) & \boldsymbol{u}(i-1) & \ldots & \boldsymbol{u}(i-M+1) \end{bmatrix}$$

and its output is given by

$$\boldsymbol{d}(i) = \boldsymbol{u}_i c + \boldsymbol{v}(i) \tag{10.17}$$

where the column vector c denotes the channel impulse response sequence, and $\boldsymbol{v}(i)$ is a zero-mean noise sequence that is uncorrelated with $\boldsymbol{u}_i$. We again remind the reader of our notation for indexing vectors and scalars in this book. We use subscripts as time indices for vectors, e.g., $\boldsymbol{u}_i$, and parenthesis as time indices for scalars, e.g., $\boldsymbol{u}(i)$.

It is further assumed that the moments $R_u = \mathsf{E}\,\boldsymbol{u}_i^*\boldsymbol{u}_i$, $\sigma_d^2 = \mathsf{E}\,|\boldsymbol{d}(i)|^2$, and $R_{du} = \mathsf{E}\,\boldsymbol{d}(i)\boldsymbol{u}_i^*$ are all independent of time. We also let $\{u_i, d(i)\}$ denote observed values for the random variables $\{\boldsymbol{u}_i, \boldsymbol{d}(i)\}$. It is important to distinguish between a measured value $d(i)$ and its stochastic version $\boldsymbol{d}(i)$; similarly for u_i and $\boldsymbol{u}_i$. This distinction is relevant because while an adaptive filtering implementation operates on the measured data $\{d(i), u_i\}$, its derivation and performance are characterized in terms of the statistical properties of the underlying stochastic variables $\{\boldsymbol{d}(i), \boldsymbol{u}_i\}$.

The channel vector c is modeled as an unknown constant vector. This situation is identical to the scenario discussed in Sec. 6.3, where c was estimated by formulating a constrained linear estimation problem — see (6.18). In terms of the notation of the present section, the rows of the data matrix H from Eq. (6.18) would be the $\{u_i\}$, while the entries of the observation vector $\boldsymbol{y}$ in (6.18) would be the $\{\boldsymbol{d}(i)\}$. The solution method (6.18) did not require knowledge of $\{R_{du}, R_u\}$. There is an alternative way to estimate c that does not require knowledge of $\{R_{du}, R_u\}$ either; it relies on using a stochastic-gradient (or adaptive) algorithm. The method can be motivated as follows. Assume we formulate the following linear least-mean-squares estimation problem:

$$\min_{w} \mathsf{E}\,|\boldsymbol{d}(i) - \boldsymbol{u}_i w|^2$$

whose solution we already know is

$$w^o = R_u^{-1} R_{du} \tag{10.18}$$

Then w^o coincides with the desired unknown c. This is because if we multiply (10.17) by $\boldsymbol{u}_i^*$ from the left and take expectations, we find that

$$\mathsf{E}\,\boldsymbol{u}_i^*\boldsymbol{d}(i) = (\mathsf{E}\,\boldsymbol{u}_i^*\boldsymbol{u}_i)\cdot c + \underbrace{\mathsf{E}\,\boldsymbol{u}_i^*\boldsymbol{v}(i)}_{=0}$$

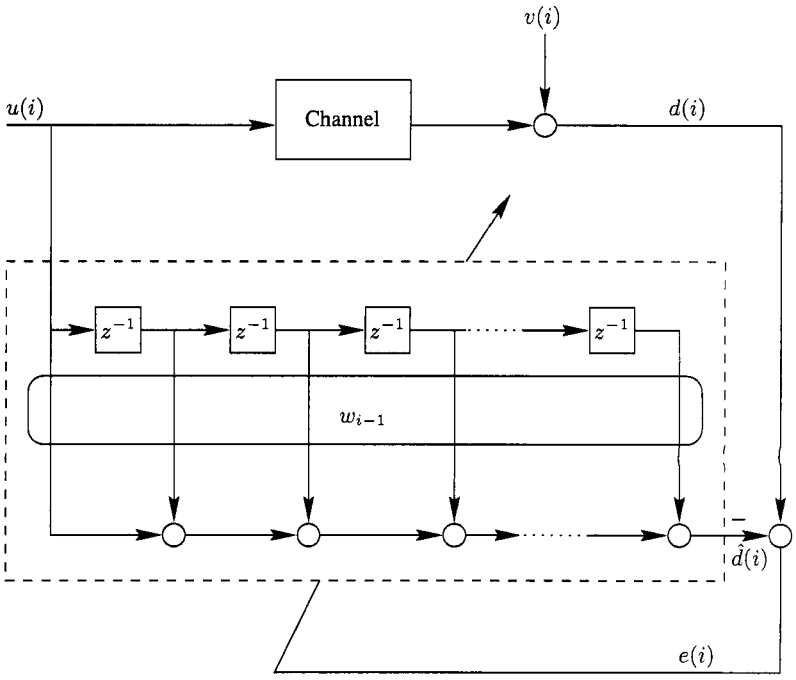

FIGURE 10.2 A structure for adaptive channel estimation.

so that $c = R_u^{-1} R_{du} = w^o$. Therefore, if we can determine w^o then we recover c. However, the moments $\{R_{du}, R_u\}$ are rarely available in practice so that determining w^o via (10.18), or even via a related steepest-descent implementation such as

$$w_i = w_{i-1} + \mu[R_{du} - R_u w_{i-1}]$$

would not be viable. Instead, we can appeal to a stochastic-gradient approximation for estimating w^o. Using measurements $\{d(i), u_i\}$, we can estimate w^o (and, hence, c) by using, e.g., the LMS recursion:

$$w_i = w_{i-1} + \mu u_i^*[d(i) - u_i w_{i-1}] \tag{10.19}$$

This discussion suggests the structure shown in Fig. 10.2, which employs an FIR filter with adjustable weights that is connected to the input and output signals $\{d(i), u(i)\}$ of the channel. At each time instant i, the measured output of the channel, $d(i)$, is compared with the output of the adaptive filter, $\hat{d}(i) = u_i w_{i-1}$, and an error signal, $e(i) = d(i) - u_i w_{i-1}$, is generated. The error is then used to adjust the filter coefficients from w_{i-1} to w_i according to (10.19). In steady-state, if the step-size μ is suitably chosen to guarantee filter convergence (usually, small step-sizes will do), then the error signal will assume small values, and the output $\hat{d}(i) = u_i w_{i-1}$ of the adaptive filter will assume values close to $d(i)$. Consequently, from an input/output perspective, the (adaptive filter) mapping from $u(i)$ to $\hat{d}(i)$ will behave similarly to the channel, which maps $u(i)$ to $d(i)$. This construction assumes that the adaptive filter has at least as many taps as the channel itself; otherwise, performance degradation can occur due to under-modeling.

Our second application is linear channel equalization. In Fig. 10.3, data symbols $\{\boldsymbol{s}(\cdot)\}$ are transmitted through a channel and the output sequence is measured in the presence of additive noise, $\boldsymbol{v}(i)$. The signals $\{\boldsymbol{v}(\cdot), \boldsymbol{s}(\cdot)\}$ are assumed uncorrelated. The noisy output of the channel is denoted by $\boldsymbol{u}(i)$ and is fed into a linear equalizer with L taps. At any particular time instant i, the state of the equalizer is given by

$$\boldsymbol{u}_i = \begin{bmatrix} \boldsymbol{u}(i) & \boldsymbol{u}(i-1) & \ldots & \boldsymbol{u}(i-L+1) \end{bmatrix}$$

It is desired to determine the equalizer tap vector w in order to estimate the signal $\boldsymbol{d}(i) = \boldsymbol{s}(i-\Delta)$ optimally in the least-mean-squares sense. This application coincides with the one described in Sec. 5.4 except that now, in conformity with the notation of this chapter, we are denoting the input to the equalizer by $\boldsymbol{u}(i)$ and the symbol to be estimated by $\boldsymbol{d}(i)$.

Clearly, the equalizer w^o that solves

$$\min_{w} \mathsf{E}\,|\boldsymbol{d}(i) - \boldsymbol{u}_i w|^2$$

is given by

$$w^o = R_u^{-1} R_{du} \tag{10.20}$$

where $R_u = \mathsf{E}\,\boldsymbol{u}_i^* \boldsymbol{u}_i$ and $R_{du} = \mathsf{E}\,\boldsymbol{d}(i)\boldsymbol{u}_i^*$. In Sec. 5.4 we assumed knowledge of the channel tap vector c (assumed FIR) and used it to evaluate $\{R_{du}, R_u\}$ — refer to equations (5.15), which were defined in terms of a channel matrix H. However, in practice, knowledge of $\{R_{du}, R_u\}$ and even c cannot be taken for granted. Actually, more often than not, these quantities are not available. For this reason, determining w^o via (10.20), or even via a related steepest-descent implementation such as

$$w_i = w_{i-1} + \mu[R_{du} - R_u w_{i-1}]$$

may not be viable. In such situations, we can appeal to a stochastic-gradient approximation for estimating w^o. Assuming an initial training phase in which transmitted data $\{d(i) = s(i-\Delta)\}$ are known at the receiver (i.e., at the equalizer), we can then use the available measurements $\{d(i), u_i\}$ to estimate w^o iteratively by using, e.g., the LMS recursion:

$$w_i = w_{i-1} + \mu u_i^*[d(i) - u_i w_{i-1}], \qquad d(i) = s(i-\Delta) \tag{10.21}$$

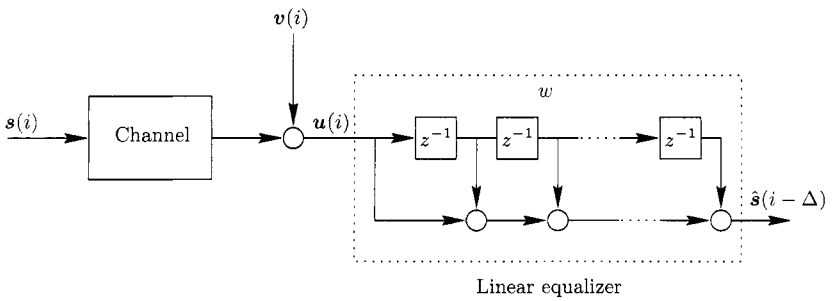

FIGURE 10.3 Linear equalization of an FIR channel in the presence of additive noise.

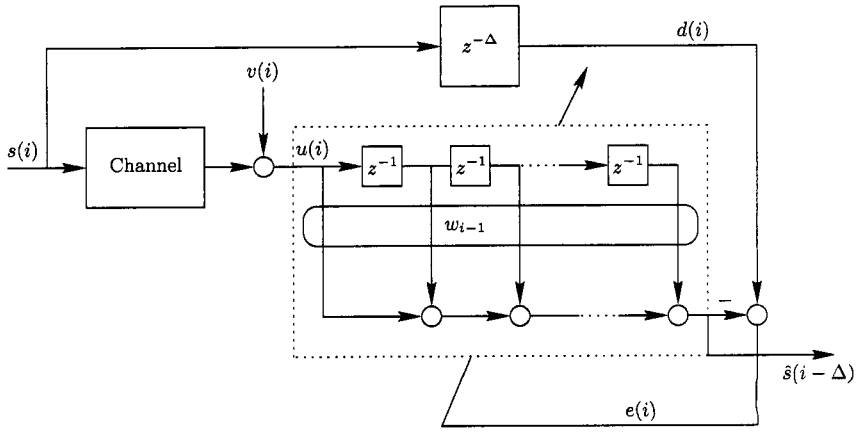

FIGURE 10.4 Adaptive linear equalization of a channel in the presence of additive noise.

This discussion suggests that we consider the structure of Fig. 10.4, with an FIR filter with adjustable weights that is connected in series with the channel. At each time instant i, the symbol $d(i) = s(i - \Delta)$ is compared with the output of the adaptive filter, $\hat{s}(i - \Delta)$, and an error signal, $e(i) = d(i) - u_i w_{i-1}$, is generated. The error is then used to adjust the filter coefficients from w_{i-1} to w_i according to (10.21). In steady-state, the error signal will assume small values and, hence, the output $\hat{s}(i - \Delta)$ of the adaptive filter will assume values close to $s(i - \Delta)$. Consequently, from an input/output perspective, the combination channel/equalizer, which maps $s(i)$ to $\hat{s}(i-\Delta)$, behaves "like" a delay system with transfer function $z^{-\Delta}$. Observe that this scheme for adaptive channel equalization does not require knowledge of the channel.

In practice, following a training phase with a known reference sequence $\{d(i)\}$, an equalizer could continue to operate in one of two modes. In the first mode, its coefficient vector w_i would be frozen and used thereafter to generate future outputs $\{\hat{s}(i - \Delta)\}$. This mode of operation is appropriate when the training phase is successful enough to result in reliable estimates $\hat{s}(i - \Delta)$, namely, estimates that lead to a low probability of error after they are fed into a decision device, which maps $\hat{s}(i - \Delta)$ to the closest point in the symbol constellation, say, $\check{s}(i - \Delta)$. However, if the channel varies slowly with time, it may be necessary to continue to operate the equalizer in a decision-directed mode. In this mode of operation, the weight vector of the equalizer continues to be adapted even after the training phase has ended. Now, however, the output of the decision device replaces the reference sequence $\{d(i)\}$ in the generation of the error sequence $\{e(i)\}$. Figure 10.5 illustrates the operation of an equalizer during the training and decision-directed modes of operation.

10.7 APPLICATION: DECISION-FEEDBACK EQUALIZATION

Our third application is decision-feedback equalization. Again, data symbols $\{\boldsymbol{s}(\cdot)\}$ are transmitted through a channel and the output sequence is measured in the presence of additive noise, $\boldsymbol{v}(i)$. The signals $\{\boldsymbol{v}(\cdot), \boldsymbol{s}(\cdot)\}$ are assumed uncorrelated. The noisy output of the channel is denoted by $\boldsymbol{u}(i)$ and is fed into a feedforward filter with L taps, as indicated

in Fig. 10.6 for the case $L = 3$. The output of the decision device is fed into a feedback filter with Q taps; this filter works in conjunction with the feedforward filter in order to supply the decision device with an estimator $\hat{\boldsymbol{s}}(i-\Delta)$. Assuming correct decisions, at any particular time instant i, the state of the equalizer is given by

$$\boldsymbol{u}_i = \left[\begin{array}{ccc|ccc} \boldsymbol{s}(i-\Delta-1) & \ldots & \boldsymbol{s}(i-\Delta-Q) & \boldsymbol{u}(i) & \ldots & \boldsymbol{u}(i-L+1) \end{array} \right]$$

That is, it contains the states of both the feedback and the feedforward filter. It is then desired to determine an equalizer tap vector

$$w \triangleq \text{col}\{-b(1), -b(2), \ldots, -b(Q), f(0), f(1), \ldots, f(L-1)\}$$

that estimates $\boldsymbol{d}(i) = \boldsymbol{s}(i-\Delta)$ optimally in the least-mean-squares sense, where the $\{-b(i)\}$ denote the coefficients of the feedback filter while the $\{f(i)\}$ denote the coefficients of the forward filter. This application coincides with the one described in Sec. 6.4, except that now, in conformity with the notation of this chapter, we are denoting the input to the equalizer by $\boldsymbol{u}(i)$ and the symbol to be estimated by $\boldsymbol{d}(i)$. We are also relying on the formulation of the decision-feedback equalizer as a linear estimation problem (cf. Eq. (6.35)).

Clearly, the equalizer w^o that solves $\min_w \mathsf{E}\,|\boldsymbol{d}(i) - \boldsymbol{u}_i w|^2$ is

$$w^o = R_u^{-1} R_{du} \tag{10.22}$$

where $R_u = \mathsf{E}\,\boldsymbol{u}_i^* \boldsymbol{u}_i$ and $R_{du} = \mathsf{E}\,\boldsymbol{d}(i)\boldsymbol{u}_i^*$. In Sec. 6.4, and also Prob. II.41, we assumed knowledge of the channel tap vector c (assumed FIR) and used it to evaluate $\{R_{du}, R_u\}$ — refer to equations (6.33) and (6.34), which were defined in terms of a channel matrix H. However, in practice, knowledge of $\{R_{du}, R_u, c\}$ is generally unavailable. For this reason, determining w^o via (10.22), or even via a related steepest-descent implementation such as

$$w_i = w_{i-1} + \mu[R_{du} - R_u w_{i-1}]$$

would not be viable. Instead, we can appeal to a stochastic-gradient approximation for estimating w^o. Assuming an initial training phase in which transmitted data $\{d(i) = s(i-\Delta)\}$

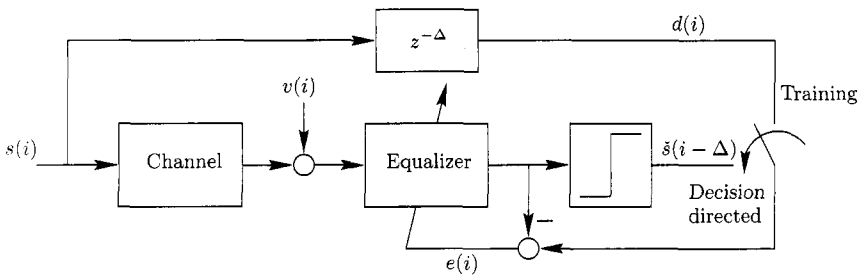

FIGURE 10.5 An adaptive linear equalizer operating in two modes: training mode and decision-direction mode.

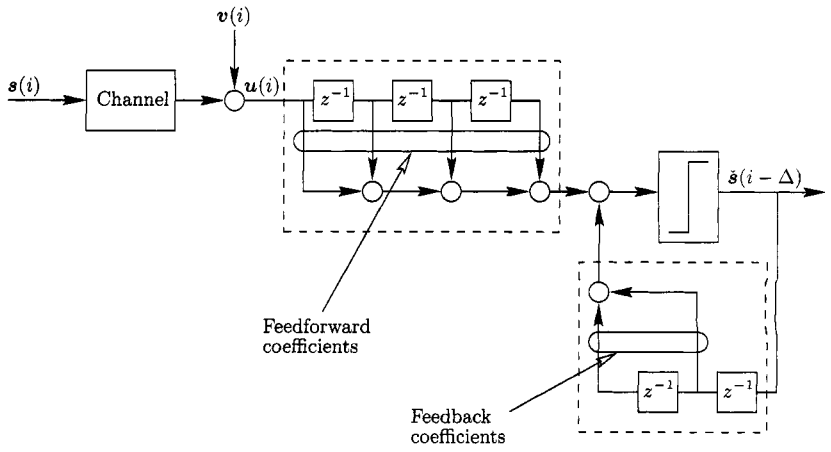

FIGURE 10.6 A decision-feedback equalizer. It consists of a feedforward filter, a feedback filter, and a decision device.

are known at the receiver (i.e., at the equalizer), we can then use the available measurements $\{d(i), u_i\}$ to estimate w^o iteratively by using, e.g., the LMS recursion:

$$w_i = w_{i-1} + \mu u_i^*[d(i) - u_i w_{i-1}], \quad d(i) = s(i-\Delta) \tag{10.23}$$

Thus, consider the structure shown in Fig. 10.7, with FIR filters with adjustable weights used as feedforward and feedback filters. The top entries of w_i would correspond to the coefficients of the feedback filter, while the bottom entries of w_i would correspond to the coefficients of the feedforward filter. Likewise, the leading entries of the regressor correspond to the state of the feedback filter, while the trailing entries of the regressor correspond to the state of the feedforward filter.

Figure 10.7 depicts two modes of operation: training and decision-directed. During training, delayed symbols are used as a reference sequence. At each time instant i, the symbol $d(i) = s(i-\Delta)$ is compared with the output of the adaptive filter, $\hat{s}(i-\Delta)$ (which is the input to the decision device), and an error signal, $e(i) = d(i) - u_i w_{i-1}$, is generated. The error is then used to adjust the coefficients of the feedback and feedforward filters according to (10.23). During training, the state (or regressor) of the equalizer is given by

$$u_i = \left[\; s(i-\Delta-1) \quad \ldots \quad s(i-\Delta-Q) \;\middle|\; u(i) \quad \ldots \quad u(i-L+1) \;\right]$$

while during decision-directed operation, the signal $d(i)$ is replaced by the output of the decision device, $\check{s}(i-\Delta)$, so that the state of the equalizer is then given by

$$u_i = \left[\; \check{s}(i-\Delta-1) \quad \ldots \quad \check{s}(i-\Delta-Q) \;\middle|\; u(i) \quad \ldots \quad u(i-L+1) \;\right]$$

10.8 ENSEMBLE-AVERAGE LEARNING CURVES

It is often necessary to evaluate the performance of a stochastic-gradient algorithm. A common way to do so is to construct its *ensemble-average* learning curve, which is defined below.

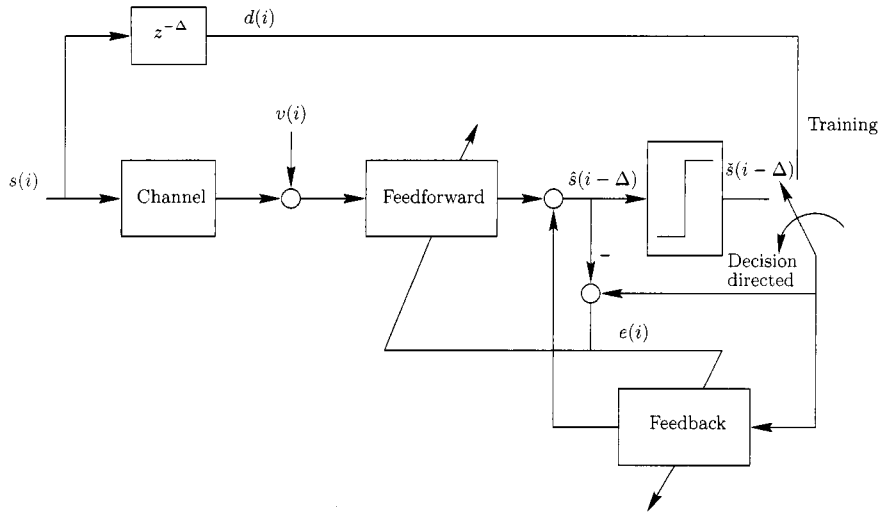

FIGURE 10.7 Adaptive decision-feedback equalization. Both modes of operation are shown: training and decision-directed operation.

Recall that for least-mean-squares estimation, the learning curve of a steepest-descent method was defined in Sec. 9.5 as (cf. (9.1)):

$$\boxed{J(i) \triangleq \mathsf{E}\,|\boldsymbol{d} - \boldsymbol{u}w_{i-1}|^2, \quad i \geq 0}$$

where w_{i-1} is the weight estimate at iteration $i-1$ that is given by the steepest-descent algorithm. Evaluation of $J(i)$ would require knowledge of $\{\sigma_d^2, R_{du}, R_u\}$. However, in a stochastic-gradient implementation, we do not have access to this statistical information but only to observations of the random variables $\boldsymbol{d}$ and $\boldsymbol{u}$, namely $\{d(i), u_i\}$. If we replace $\boldsymbol{d}$ by $d(i)$ and $\boldsymbol{u}$ by u_i in the above expression for $J(i)$, then the difference $\boldsymbol{d} - \boldsymbol{u}w_{i-1}$ becomes $d(i) - u_i w_{i-1}$, with the w_{i-1} now denoting the weight estimate that is obtained from the stochastic-gradient implementation (e.g., LMS). We have denoted the difference $d(i) - u_i w_{i-1}$ by $e(i)$ earlier in this chapter and called it the *a priori* output estimation error,

$$\boxed{e(i) = d(i) - u_i w_{i-1}}$$

We can then estimate the learning curve of an adaptive filter as follows. We run the algorithm for a certain number of iterations, say, for $0 \leq i \leq N$. The duration N is usually chosen large enough so that convergence is observed. We then compute the error sequence $\{e(i)\}$ and the corresponding squared-error curve $\{|e(i)|^2,\ 0 \leq i \leq N\}$. We denote this squared-error curve by

$$\left\{\left|e^{(1)}(i)\right|^2,\ 0 \leq i \leq N\right\}$$

with the superscript $^{(1)}$ used to indicate that this curve is the result of the first experiment. We then run the same stochastic-gradient algorithm a second time, starting from the same

initial condition w_{-1} and using data with the same statistical properties as in the first run. After L such experiments, we obtain L squared-error curves,

$$\left\{ \left|e^{(1)}(i)\right|^2, \quad \left|e^{(2)}(i)\right|^2, \ \ldots, \quad \left|e^{(L)}(i)\right|^2 \right\}, \quad 0 \leq i \leq N$$

The *ensemble-average* learning curve, over the interval $0 \leq i \leq N$, is defined as the average over the L experiments:

$$\boxed{\widehat{J}(i) \triangleq \frac{1}{L}\left(\sum_{j=1}^{L} \left|e^{(j)}(i)\right|^2\right), \quad i \geq 0}$$

The averaged curve $\widehat{J}(i)$ so defined is a sample-average approximation of the true learning curve $J(i)$.

Example 10.1 (Learning curves)

We illustrate the construction of learning curves by considering an example in the context of channel estimation, as described in Sec. 10.5. The impulse response sequence of the channel is chosen as $c = \text{col}\{1, 0.5, -1, 2\}$, i.e., its transfer function is

$$C(z) = 1 + 0.5z^{-1} - z^{-2} + 2z^{-3}$$

The channel impulse response, along with its magnitude frequency response, are shown in Fig. 10.8.

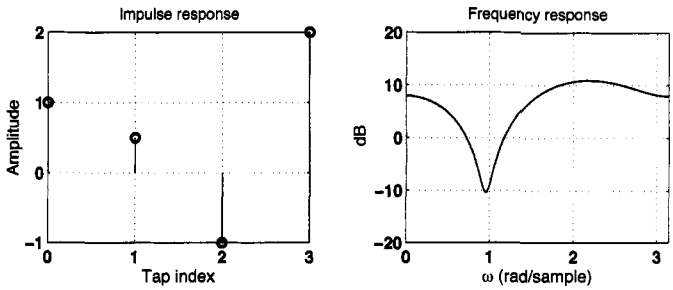

FIGURE 10.8 The impulse-response sequence and the magnitude-frequency response of the channel $C(z) = 1 + 0.5z^{-1} - z^{-2} + 2z^{-2}$.

White input data $\{u(i)\}$ of unit variance is fed into the channel and the output sequence is observed in the presence of white additive noise of variance 0.01. A total of $N = 600$ samples $\{u(i), d(i)\}$ are generated and used to train an adaptive filter with $M = 4$ taps. The filter is trained by using the LMS algorithm of this chapter with step-size $\mu = 0.01$, as well as two other algorithms derived in Chapters 11 and 14 for comparison purposes, namely, the so-called ϵ−NLMS algorthm with step-size $\mu = 0.2$ and $\epsilon = 0.001$, and the RLS algorithm with $\lambda = 0.995$ and the same value of ϵ. All filters start from the same initial condition $w_{-1} = 0$.

Figure 10.9 shows two typical instantaneous squared-error curves for each of the algorithms over the first 200 iterations, i.e., the rows show plots of $|e(i)|^2$ versus time in two random simulations for each algorithm. Observe how the curves die out quicker for ϵ−NLMS and RLS relative to LMS. By averaging $L = 300$ such curves for each algorithm, we obtain the ensemble-averaging learning curves shown in Fig. 10.10. These curves illustrate the fact that the convergence speed increases as we move from LMS to ϵ−NLMS to RLS.

◇

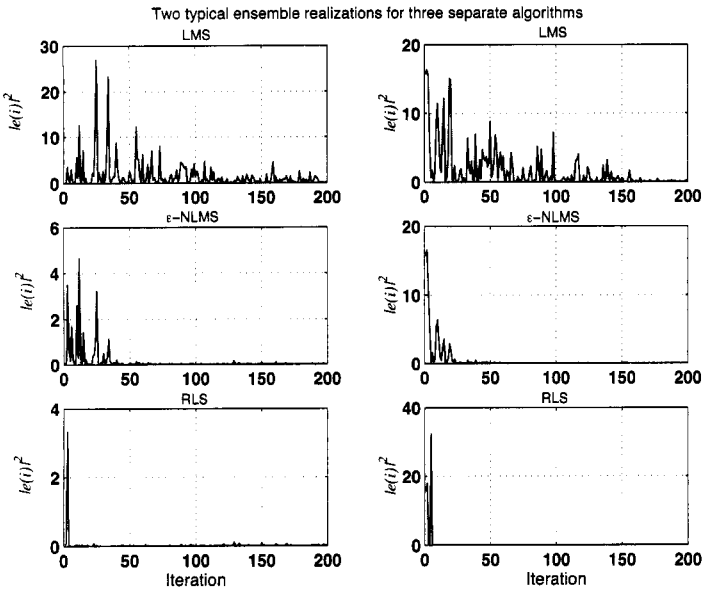

FIGURE 10.9 Typical squared-error curves for LMS, ϵ-NLMS, and RLS.

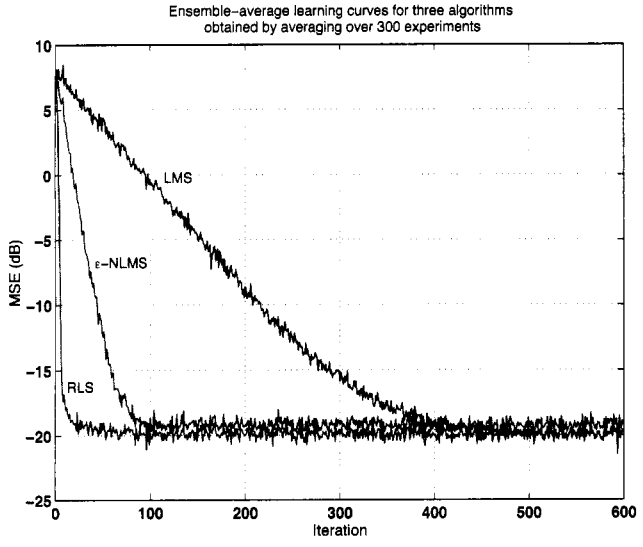

FIGURE 10.10 Ensemble-average learning curves for LMS, ϵ-NLMS, and RLS obtained by averaging over 300 experiments.