1 Introduction

1.1 Introduction

Adaptive filtering has been a widely utilized signal processing tool, and is the subject of many textbooks including [1–9] just to mention a few. Usually, the classical adaptive filtering algorithms belong to the class of supervised learning algorithms, although there are unsupervised versions of them, such as the blind adaptive filters. In the supervised case, filtering structures tend to be quite simple, allowing the use of more complex learning algorithms. In the latter case, filtering structures are usually nonlinear and can be quite complex so that employing sophisticated learning algorithms might not be possible.

This book aims to describe more recent developments in this field not fully addressed in the classical texts. This chapter summarizes the main equations and features related to the most popular adaptive filtering algorithms. Often, these algorithms serve as a basis to the algorithms introduced in the subsequent chapters. In all cases, we describe the actual objective function utilized by each algorithm so that, from the engineering point of view, one can get a grasp of what the algorithms are minimizing. In addition to that, this Introduction establishes the notation used in this book. However, an experienced researcher in adaptive filtering may choose to skip this chapter without further consequences.

1.2 Data Description

The most common adaptive filtering configuration belongs to the class of supervised learning where an input signal is transformed into an output signal that in turn tries to track the behavior of a reference signal. Figure 1.1 depicts the typical adaptive filtering setup.

In most cases, the signals to be processed by an adaptive filter consist of an input signal denoted by $x(k)$ and a reference or desired signal denoted by $d(k)$. The input signal in the simplest environment is collected in a delay line vector as

$$\mathbf{x}(k) = [x(k)\, x(k-1)\, \cdots\, x(k-N)]^{\mathrm{T}}, \tag{1.1}$$

https://doi.org/10.1017/9781108896139.002 Published online by Cambridge University Press

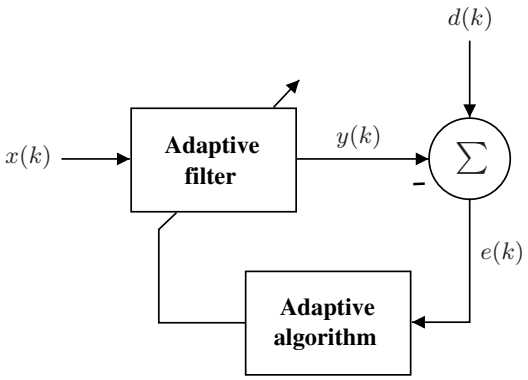

Figure 1.1 Adaptive filtering configuration.

where vector $\mathbf{x}(k) \in \mathbb{C}^{(N+1)\times 1}$, with $\mathbb{C}$ representing the complex numbers. In many situations, $\mathbf{x}(k) \in \mathbb{R}^{(N+1)\times 1}$, where $\mathbb{R}$ represents the real numbers.

The input-signal vector generates the output signal of the adaptive filter through the inner product, also known as the tapped delay line,

$$y(k) = \mathbf{w}^{\mathrm{H}}(k)\mathbf{x}(k), \tag{1.2}$$

where $\mathbf{w}(k) = [w_0(k)\ w_1(k)\ \cdots\ w_N(k)]^{\mathrm{T}}$ represents the adaptive filter coefficients or parameter vector, and $y(k)$ is the adaptive filter output. The latter output signal is compared with the reference signal to form the (*a priori*) error signal as follows:

$$e(k) = d(k) - \mathbf{w}^{\mathrm{H}}(k)\mathbf{x}(k). \tag{1.3}$$

The adaptive filter minimizes a cost function, also known as the objective function, of the error signal represented as

$$\xi(k) = \mathbb{F}[e(k)], \tag{1.4}$$

with $\mathbb{F}[\cdot]$ assuming distinct definitions depending on the algorithm and application.

1.3 The LMS Algorithm

The least mean squares (LMS) algorithm is recognized as the most widely used adaptive filtering algorithm ever, for its implementation simplicity and good performance in many practical situations. The LMS origins are described in the review chapters [10, 11], and its performance and features are addressed in many textbooks [1–9].

The basic concept behind the LMS algorithm comes from the steepest descent approach [1], where the coefficient vector is updated in the opposite direction

https://doi.org/10.1017/9781108896139.002 Published online by Cambridge University Press

of the gradient $\nabla_{\mathbf{W}^*}\xi(k)$, where $\xi(k) = \mathbb{E}[|d(k) - \mathbf{w}^{\mathrm{H}}\mathbf{x}(k)|^2]$ is the mean square error (MSE). In fact, the LMS algorithm uses an instantaneous estimate of the gradient at instant k, with the simplest possible option given by

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu\nabla_{\mathbf{W}^*}\hat{\xi}(k), \tag{1.5}$$

where $\mu > 0$, known as step size, is responsible for controlling the convergence of the algorithm and $\hat{\xi}(k) = |e(k)|^2$. Another, and more elegant, way to obtain the coefficient update of the LMS algorithm comes from minimizing the following cost function:

$$\begin{aligned}\xi_{\mathrm{LMS}}(k) &= \|\mathbf{w}(k+1) - \mathbf{w}(k)\|^2_{\mathbf{P}(k)} + \mu|\varepsilon(k)|^2 \\ &= \|\mathbf{w}(k+1) - \mathbf{w}(k)\|^2_{\mathbf{P}(k)} + \mu|d(k) - \mathbf{w}^{\mathrm{H}}(k+1)\mathbf{x}(k)|^2,\end{aligned} \tag{1.6}$$

where the weighted quadratic norm is defined as $\|\mathbf{x}\|^2_{\mathbf{P}(k)} = \mathbf{x}^{\mathrm{H}}\mathbf{P}(k)\mathbf{x}$ and $\mathbf{P}(k) = \mathbf{I} - \mu\mathbf{x}(k)\mathbf{x}^{\mathrm{H}}(k)$, with μ being small enough to guarantee that $\mathbf{P}(k)$ is positive definite. It includes two terms: one related to the disturbance on the coefficients $\|\mathbf{w}(k+1) - \mathbf{w}(k)\|^2$ and the other related to the instantaneous squared *a posteriori* error $|\varepsilon(k)|^2$. It is worth mentioning that in Equation (1.6) $\mu \leq \frac{1}{\|\mathbf{x}\|^2}$ to keep the cost function convex.

By computing the gradient of the objective function with respect to $\mathbf{w}(k+1)$, the coefficient update that minimizes the objective function is

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu e^*(k)\mathbf{x}(k). \tag{1.7}$$

As can be seen, the coefficient update equation of the LMS algorithm is quite simple and requires low computational costs. There is a vast literature addressing several issues pertaining to the LMS algorithm for which the previously mentioned references [1–9] represent a tiny sample.

1.4 The RLS Algorithm

The standard recursive least squares (RLS) algorithm minimizes the following function [1, 12, 13]:

$$\begin{aligned}\xi^{\mathrm{d}}(k) &= \sum_{i=0}^{k} \lambda^{k-i}|\varepsilon(i)|^2 \\ &= \sum_{i=0}^{k} \lambda^{k-i}\left|d(i) - \mathbf{w}^{\mathrm{H}}(k+1)\mathbf{x}(i)\right|^2,\end{aligned} \tag{1.8}$$

where $\varepsilon(i)$ is defined as the *a posteriori* error measured with the data entries $d(i)$ and $\mathbf{x}(i)$, and λ is the forgetting factor parameter. Another way to express the objective function is

https://doi.org/10.1017/9781108896139.002 Published online by Cambridge University Press

$$\xi^{\mathrm{d}}(k) = |d(k)|^2 - 2\mathrm{Re}\left[d^*(k)\mathbf{w}^{\mathrm{H}}(k+1)\mathbf{x}(k)\right] + \left|\mathbf{w}^{\mathrm{H}}(k+1)\mathbf{x}(k)\right|^2 + \lambda\sum_{i=0}^{k-1}\lambda^{k-1-i}|\varepsilon(i)|^2, \tag{1.9}$$

where Re[·] means real part of [·] from which a recursive solution can be obtained.

As previously seen for the LMS algorithm, the RLS algorithm also has a *minimum disturbance* cost function that is given as [14]

$$\xi_{\mathrm{RLS}}(k) = \lambda\|\mathbf{w}(k+1) - \mathbf{w}(k)\|^2_{\mathbf{R}_{\mathrm{D}}(k)} + |d(k) - \mathbf{w}^{\mathrm{H}}(k+1)\mathbf{x}(k)|^2, \tag{1.10}$$

where $\mathbf{R}_{\mathrm{D}}(k) = \sum_{i=0}^{k}\lambda^{k-i}\mathbf{x}(i)\mathbf{x}^{\mathrm{H}}(i)$, whose inverse is the matrix $\mathbf{S}_{\mathrm{D}}(k)$ used in the following.

The corresponding update equations of the standard RLS algorithm consist of

$$e(k) = d(k) - \mathbf{w}^{\mathrm{H}}(k)\mathbf{x}(k), \tag{1.11}$$

$$\boldsymbol{\psi}(k) = \mathbf{S}_{\mathrm{D}}(k-1)\mathbf{x}(k), \tag{1.12}$$

$$\mathbf{S}_{\mathrm{D}}(k) = \frac{1}{\lambda}\left[\mathbf{S}_{\mathrm{D}}(k-1) - \frac{\boldsymbol{\psi}(k)\boldsymbol{\psi}^{\mathrm{H}}(k)}{\lambda + \boldsymbol{\psi}^{\mathrm{H}}(k)\mathbf{x}(k)}\right], \text{ and} \tag{1.13}$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + e^*(k)\mathbf{S}_{\mathrm{D}}(k)\mathbf{x}(k), \tag{1.14}$$

where an initialization is required for the following quantities: $\mathbf{S}_{\mathrm{D}}(-1) = \rho\mathbf{I}$, ρ being the inverse of an estimate of the input signal power times $1-\lambda$, and $\mathbf{x}(-1) = \mathbf{w}(0) = [0\,0\cdots0]^{\mathrm{T}}$.

In a more general RLS algorithm, employing a time-varying forgetting factor, the objective function we seek to minimize is given by

$$\xi^{\mathrm{d}}(k) = \sum_{i=0}^{k}\lambda^{k-i+1}(i)|\varepsilon(i)|^2 = \sum_{i=0}^{k}\lambda^{k-i+1}(i)\left|d(i) - \mathbf{w}^{\mathrm{H}}(k+1)\mathbf{x}(i)\right|^2, \tag{1.15}$$

where $\varepsilon(i)$ is also the *a posteriori* error measured with the comparison between $d(i)$ and $\mathbf{w}^{\mathrm{H}}(k+1)\mathbf{x}(i)$. The cost function can be written in an alternative form as

$$\xi^{\mathrm{d}}(k) = \lambda(k)\left[|d(k)|^2 - 2\mathrm{Re}\left[d^*(k)\mathbf{w}^{\mathrm{H}}(k+1)\mathbf{x}(k)\right] + \left|\mathbf{w}^{\mathrm{H}}(k+1)\mathbf{x}(k)\right|^2\right] + \sum_{i=0}^{k-1}\lambda^{k-i+1}(i)|\varepsilon(i)|^2. \tag{1.16}$$

The basic update equations of the generalized RLS algorithm are given by the following relations:

https://doi.org/10.1017/9781108896139.002 Published online by Cambridge University Press

$$e(k) = d(k) - \mathbf{w}^{\mathrm{H}}(k)\mathbf{x}(k), \tag{1.17}$$

$$\boldsymbol{\psi}(k) = \mathbf{S}_{\mathrm{D}}(k-1)\mathbf{x}(k), \tag{1.18}$$

$$\mathbf{S}_{\mathrm{D}}(k) = \mathbf{S}_{\mathrm{D}}(k-1) - \frac{\lambda(k)\boldsymbol{\psi}(k)\boldsymbol{\psi}^{\mathrm{H}}(k)}{1+\lambda(k)\boldsymbol{\psi}^{\mathrm{H}}(k)\mathbf{x}(k)}, \text{ and} \tag{1.19}$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \lambda(k)e^{*}(k)\mathbf{S}_{\mathrm{D}}(k)\mathbf{x}(k). \tag{1.20}$$

In the above relations, the initialization entails choosing $\mathbf{S}_{\mathrm{D}}(-1) = \rho\mathbf{I}$, where ρ can be the inverse of an estimate of the input signal power times $1-\lambda(0)$, and $\mathbf{x}(-1) = \mathbf{w}(0) = [0\,0\cdots 0]^{\mathrm{T}}$.

In the standard RLS algorithm, the value of the forgetting factor is usually chosen as $0 \ll \lambda \leq 1$, whereas in the generalized RLS $0 \ll \lambda(k) \leq 1$ and the choice of $\lambda(k)$ depends on the cost function. The latter case encompasses algorithms like BEACON versions of [17, 18] and of some references therein.

1.5 Affine Projection Algorithms

The affine projection (AP) algorithm assembles and reuses the last $L+1$ input signal vectors in an input-vector matrix according to [1, 15]:

$$\mathbf{X}(k) = [\mathbf{x}(k)\,\mathbf{x}(k-1)\cdots\mathbf{x}(k-L)], \tag{1.21}$$

where $\mathbf{X}(k) \in \mathbb{C}^{(N+1)\times(L+1)}$. At iteration k, we define the desired signal vector and the error vector, respectively, represented by

$$\mathbf{d}(k) = [d(k)\;d(k-1)\;\ldots\;d(k-L)]^{\mathrm{T}} \text{ and} \tag{1.22}$$

$$\mathbf{e}(k) = [e_0(k)\;e_1(k)\;\cdots\;e_L(k)]^{\mathrm{T}}, \tag{1.23}$$

where vectors $\mathbf{e}(k) \in \mathbb{C}^{(L+1)\times 1}$ and $\mathbf{d}(k) \in \mathbb{C}^{(L+1)\times 1}$ retain information from the $L+1$ last iterations. The entries of the error vector are defined as $e_i(k) = d(k-i) - \mathbf{w}^{\mathrm{H}}(k)\mathbf{x}(k-i)$, for $i \in \{0, 1, \ldots, L\}$.

The AP algorithm presented here minimizes the objective function

$$\xi_{\mathrm{AP}}(k) = \|\mathbf{w}(k+1) - \mathbf{w}(k)\|^2_{\mathbf{P}(k)} + \|\mathbf{d}(k) - \mathbf{X}^{\mathrm{T}}(k)\mathbf{w}^{*}(k+1)\|^2_{\mathbf{A}(k)\mathbf{F}}, \tag{1.24}$$

where

$$\mathbf{A}(k) = \left(\mathbf{X}^{\mathrm{H}}(k)\mathbf{X}(k) + \gamma\mathbf{I}\right)^{-1} \text{ and} \tag{1.25}$$

$$\mathbf{P}(k) = \frac{1}{\mu}\mathbf{I} - \mathbf{X}(k)\mathbf{A}(k)\mathbf{F}\mathbf{X}^{\mathrm{H}}(k), \tag{1.26}$$

where μ represents the step size (or a learning factor) of the adaptive algorithm and γ is the regularization factor introduced to avoid numerical problems when matrix $\left(\mathbf{X}^{\mathrm{H}}(k)\mathbf{X}(k)\right)^{-1}$ becomes ill conditioned. The value of μ is small to keep the cost function convex. Matrix $\mathbf{F}$ might assume different forms depending on the desired characteristics of the adaptive algorithm and the affordable computational complexity [19–21].

https://doi.org/10.1017/9781108896139.002 Published online by Cambridge University Press

The cost function in Equation (1.24) has two terms, where the first one implements the minimum disturbance in the adaptive coefficients weighted by matrix $\mathbf{P}(k)$, and the Euclidean norm of the *a posteriori* error vector properly normalized through matrix $\mathbf{A}(k)$. Matrices $\mathbf{A}(k)$ and $\mathbf{P}(k)$ are positive definite.

The standard AP algorithm utilizes $\mathbf{F} = \mathbf{I}$, leading to the coefficient updating in the form

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu \mathbf{X}(k) \left(\mathbf{X}^{\mathrm{H}}(k)\mathbf{X}(k) + \gamma \mathbf{I} \right)^{-1} \mathbf{e}^{*}(k). \tag{1.27}$$

1.6 The Normalized LMS Algorithm

The NLMS algorithm, proposed in 1967 [22, 23] and widely used in practice, came as a response to the main drawbacks of the LMS algorithm, previously introduced in 1960 [24]: the slow convergence for correlated input signals and the need to choose an appropriate step size. One can note that the LMS algorithm updates in the direction of the input vector $\mathbf{x}(k)$. If we assume perfect modeling and no observation error, we could figure out that, given the data pair $d(k)$ and $\mathbf{x}(k)$, the optimal point $\mathbf{w}_{\mathrm{o}}$ would belong to a hyperplane $\mathbf{w}^{\mathrm{H}}\mathbf{x}(k) = d(k)$, which happens to be orthogonal to the input vector $\mathbf{x}(k)$. Therefore, as shown in Figure 1.2, we could choose a step size that leads to a null *a posteriori* error. The variable step size can be easily obtained, and its value corresponds to $\frac{1}{\mathbf{x}^{\mathrm{H}}(k)\mathbf{x}(k)}$. We may also obtain the updating expression of the NLMS algorithm by minimizing the following cost function:

$$\xi_{\mathrm{NLMS}}(k) = \|\mathbf{w}(k+1) - \mathbf{w}(k)\|^2_{\mathbf{P}(k)} \text{ subject to } \mathbf{w}^{\mathrm{H}}(k+1)\mathbf{x}(k) = d(k). \tag{1.28}$$

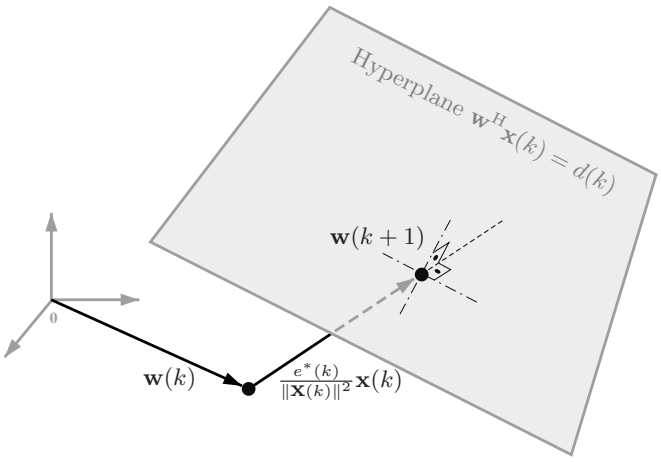

Figure 1.2 Coefficient vector updating of the NLMS algorithm with null *a posteriori* error.

https://doi.org/10.1017/9781108896139.002 Published online by Cambridge University Press

The resulting updating expression of the NLMS algorithm is given as follows, where another step size was added to allow a trade-off between speed of convergence and misadjustment:[1]

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu \frac{e^*(k)}{\mathbf{x}^{\mathrm{H}}(k)\mathbf{x}(k) + \gamma}\mathbf{x}(k), \tag{1.29}$$

where γ is a small positive number added to avoid divisions by 0.

Although the NLMS step size can theoretically range from 0 to 2, we recommend its use within $0 < \mu \leq 1$. Since the fastest convergence is obtained with $\mu = 1$, using a larger value would increase misadjustment and decrease convergence speed. Although developed independently and with another approach, the AP algorithm with the particular case where $L = 0$ becomes the NLMS algorithm.

1.7 Set-Membership Affine Projection Algorithms

Set-membership filtering (SMF) is a noteworthy example of connecting set-theoretic estimation with data selection, enabling a reduction of computational burden and, consequently, energy savings. As a bonus, in addition to reducing computational burden, SMF-based algorithms tend to be more robust against noise [16, 29].

The SMF concept appeared in [25] and is suitable for linear adaptive filtering problems. Its criterion aims to estimate the vector $\mathbf{w}$ that leads to an error signal $e = d - \mathbf{w}^{\mathrm{H}}\mathbf{x} \in \mathbb{C}$ whose magnitude is upper bounded by a constant $\overline{\gamma}$ for all possible pairs $\{\mathbf{x}, d\}$. Variable $\overline{\gamma}$ determines how much error is acceptable and is usually chosen based on *a priori* information about the sources of uncertainty. Most of the time, we assume that such uncertainty is caused by an observation error whose variance is σ_n^2, and $\overline{\gamma}$ is chosen as a function of σ_n^2.

Denoting $\mathcal{S}$ as the set comprising all possible pairs $\{\mathbf{x}, d\}$, we can state SMF is interested in finding $\mathbf{w}$ that satisfies $|e| = |d - \mathbf{w}^{\mathrm{H}}\mathbf{x}| \leq \overline{\gamma},\ \forall\{\mathbf{x}, d\} \in \mathcal{S}$. That is, by defining the *feasibility set* as

$$\Theta \triangleq \bigcap_{\{\mathbf{x},d\}\in\mathcal{S}} \left\{\mathbf{w} \in \mathbb{C}^{N+1} : |d - \mathbf{w}^{\mathrm{H}}\mathbf{x}| \leq \overline{\gamma}\right\}, \tag{1.30}$$

the SMF criterion can be summarized as finding a vector $\mathbf{w} \in \Theta$.

Considering online applications, the previous expression does not provide a practical way of determining Θ or a point in it since we do not have $\mathcal{S}$. The set-membership approach to iterative techniques, referred to as *set-membership*

[1] Misadjustment is defined as the ratio between the excess MSE and the minimum MSE, that is, $M = \frac{\xi(\infty) - \xi_{\min}}{\xi_{\min}}$ [1], where $\xi(\infty)$ corresponds to the steady-state MSE.

https://doi.org/10.1017/9781108896139.002 Published online by Cambridge University Press

adaptive recursive techniques [25], is an alternative adaptive filtering formulation featuring a data-selective updating. This approach maintains the concept that, if the set of parameters leads to an error magnitude below a given threshold, no coefficient update is required at that particular iteration.

Consider a set of data pairs $\{\mathbf{x}(i), d(i)\}$, for $i \in \{0, 1, \ldots, k\}$, and define $\mathcal{H}(k)$ as the set containing all vectors $\mathbf{w}$ such that the associated output error at time instant k is upper bounded in magnitude by $\bar{\gamma}$, so that,

$$\mathcal{H}(k) = \{\mathbf{w} \in \mathbb{C}^{N+1} : |d(k) - \mathbf{w}^{\mathrm{H}}\mathbf{x}(k)| \leq \bar{\gamma}\}. \tag{1.31}$$

This set $\mathcal{H}(k)$ is called the *constraint set*. In the two-dimensional case ($N = 1$), the boundaries of $\mathcal{H}(k)$ are such that the error values are $\bar{\gamma}\mathrm{e}^{\mathrm{j}\phi}$, for $\phi \in [0, 2\pi)$, that is, $\mathcal{H}(k)$ comprises the region between the lines where $\left|d(k) - \mathbf{w}^{\mathrm{H}}\mathbf{x}(k)\right| = \bar{\gamma}$. For higher dimensions, hyperplanes delimit $\mathcal{H}(k)$; see [25–27] for details and [1, 28] for further developments.

Following the formulation of Section 1.5, the set-membership affine projection (SM-AP) algorithm, when updating, minimizes the following objective function:

$$\xi_{\mathrm{sm}}(k) = \|\mathbf{w}(k+1) - \mathbf{w}(k)\|^2_{\mathbf{P}(k)} + \|\mathbf{d}(k) - \mathbf{X}^{\mathrm{T}}(k)\mathbf{w}^*(k+1) - \bar{\boldsymbol{\gamma}}(k)\|^2_{\mathbf{A}(k)\mathbf{F}}, \tag{1.32}$$

where $\bar{\gamma}_i(k)$, the entries of $\bar{\boldsymbol{\gamma}}(k)$, are chosen such that $|\bar{\gamma}_i(k)| \leq \bar{\gamma}$ for $i \in \{1, \ldots, L+1\}$.

There are many ways to choose the entries of the constraint vector as long as they correspond to points represented by the adaptive-filter coefficients in $\mathcal{H}(k - i + 1)$, that is, $|\bar{\gamma}_i(k)| \leq \bar{\gamma}$. The choice of $\bar{\boldsymbol{\gamma}}(k)$ affects the overall computational complexity. By choosing $\bar{\gamma}_1(k) = e(k)/|e(k)|$ and all the remaining elements of $\bar{\boldsymbol{\gamma}}(k)$ as zeros, the solution is called a simple choice and results in a reduced computational complexity version of the SM-AP algorithm [16, 29].

Using $\mathbf{F} = \mathbf{I}$ and the simple constraint vector, the SM-AP algorithm has the following coefficient updating form:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu(k)\mathbf{X}(k)\left(\mathbf{X}^{\mathrm{H}}(k)\mathbf{X}(k) + \gamma\mathbf{I}\right)^{-1} e^*(k)\mathbf{u}_1, \tag{1.33}$$

where $\mathbf{u}_1^{\mathrm{T}} = [1\, 0 \cdots 0]$,

$$e(k) = d(k) - \mathbf{w}^{\mathrm{H}}(k)\,\mathbf{x}(k), \text{ and} \tag{1.34}$$

$$\mu(k) = \begin{cases} 1 - \frac{\bar{\gamma}}{|e(k)|} & \text{if } |e(k)| > \bar{\gamma}, \\ 0 & \text{otherwise;} \end{cases} \tag{1.35}$$

see [1] for details.

The algorithms presented in a simplified form in this chapter will appear in modified forms in the forthcoming chapters, each one adapted to the situation at hand.

https://doi.org/10.1017/9781108896139.002 Published online by Cambridge University Press

Table 1.1 Expressions related to the misadjustment

Algorithm	M
LMS	$\frac{\mu \text{tr}[\mathbf{R}]}{1-\mu \text{tr}[\mathbf{R}]}$
NLMS	$\frac{\mu(N+2)}{(2-\mu)(N-1)}$
AP	$\frac{(L+1)\mu}{2-\mu}\frac{1-(1-\mu)^2}{1-(1-\mu)^{2(L+1)}}$
simple SM-AP	$\frac{(L+1)p_{\text{up}}}{2-p_{\text{up}}}\left(\frac{\bar{\gamma}^2}{\sigma_n^2}+1\right)$
RLS	$(N+1)\frac{1-\lambda}{2-(1-\lambda)}$

Table 1.2 Computational complexity in adaptive filtering algorithms

Algorithm	Multiplication	Addition	Division
LMS	$2N+3$	$2N+2$	0
NLMS	$2N+4$	$2N+5$	1
AP	$(2.5N+L+10)(L+1)$	$(1.5N+2.5)(L+1)+2N$	$L+2$
simple SM-AP[a]	$(1.5N+3L+7.5)(L+1)$	$(1.5N+7.5)(L+1)+0.5L$	$L+2$
RLS	$3N^2+11N+8$	$3N^2+7N+4$	1

[a] This estimate is an upper bound assuming that the updates occur all the time.

1.8 Performance and Computational Complexity

Table 1.1 lists the misadjustment expressions for the main adaptive filtering algorithms for a stationary environment. This information reveals the accuracy of the algorithms after convergence in stationary environments. The expression of the misadjustment related to the simple SM-AP algorithm includes the ratio between the threshold parameter and the background noise variance σ_n^2, as well as the probability of achieving an update, denoted as p_{up}. Reference [1] details how to derive these expressions.

As for the computational complexity, Table 1.2 lists the expressions (number of operations per update) for the classical adaptive filtering algorithms. This information is relevant to choose correctly the right family of algorithms, particularly in computationally sensitive applications.

1.9 Conclusion

This chapter introduced the classical adaptive filtering algorithms that are widely used in many applications. This presentation sets the conditions to extend and modify these algorithms to solve problems related to kernel adaptive filtering, sparsity aware learning, distributed learning, array processing, and learning on graphs. The algorithms were described in a concise format starting from their actual cost functions. We believe this is a shortcut to simplify the description of the more general algorithms addressed in the following chapters.

We summarize in Algorithm 1.1 all classical adaptive algorithms reviewed in this introduction. In the forthcoming chapters, real and complex arithmetic versions of these algorithms are utilized where appropriate.

https://doi.org/10.1017/9781108896139.002 Published online by Cambridge University Press

Algorithm 1.1 The classical algorithms

Initialization

$\mathbf{w}_{\text{LMS}}(k) = \mathbf{w}_{\text{NLMS}}(k) = \mathbf{w}_{\text{AP}}(k) = \mathbf{w}_{\text{SMAP}}(k) = \mathbf{w}_{\text{RLS}}(k) = \mathbf{0}$

$\mathbf{u}_1^{\text{T}} = [1\, 0 \ldots 0]$

For $k > 0$ **do**

The LMS Algorithm

$e(k) = d(k) - \mathbf{w}_{\text{LMS}}^{\text{H}}(k)\mathbf{x}(k)$

$\mathbf{w}_{\text{LMS}}(k+1) = \mathbf{w}_{\text{LMS}}(k) + \mu e^*(k)\mathbf{x}(k)$

The NLMS Algorithm

$e(k) = d(k) - \mathbf{w}_{\text{NLMS}}^{\text{H}}(k)\mathbf{x}(k)$

$\mathbf{w}_{\text{NLMS}}(k+1) = \mathbf{w}_{\text{NLMS}}(k) + \mu \frac{e^*(k)}{\mathbf{x}^{\text{H}}(k)\mathbf{x}(k)+\gamma}\mathbf{x}(k)$

The Affine Projections Algorithm

$\mathbf{d}(k) = [d(k)\, d(k-1)\, \cdots\, d(k-L)]^{\text{T}}$

$\mathbf{X}(k) = [\mathbf{x}(k)\, \mathbf{x}(k-1)\, \cdots\, \mathbf{x}(k-L)]$

$\mathbf{e}^*(k) = \mathbf{d}^*(k) - \mathbf{X}^{\text{H}}(k)\mathbf{w}_{\text{AP}}(k)$

$\mathbf{w}_{\text{AP}}(k+1) = \mathbf{w}_{\text{AP}}(k) + \mu\mathbf{X}(k)\left[\mathbf{X}^{\text{H}}(k)\mathbf{X}(k) + \gamma\mathbf{I}\right]^{-1}\mathbf{e}^*(k)$

The Set-Membership Affine Projections Algorithm (simple choice)

$e(k) = d(k) - \mathbf{w}_{\text{SMAP}}^{\text{H}}\mathbf{x}(k)$

$$\mu(k) = \begin{cases} 1 - \frac{\bar{\gamma}}{|e(k)|} & \text{if } |e(k)| > \bar{\gamma} \\ 0 & \text{otherwise} \end{cases}$$

$\mathbf{w}_{\text{SMAP}}(k+1) = \mathbf{w}_{\text{SMAP}}(k) + \mu(k)\mathbf{X}(k)\left(\mathbf{X}^{\text{H}}(k)\mathbf{X}(k) + \gamma\mathbf{I}\right)^{-1}e^*(k)\mathbf{u}_1$

The RLS Algorithm

$e(k) = d(k) - \mathbf{w}_{\text{RLS}}^{\text{H}}(k)\mathbf{x}(k)$

$\boldsymbol{\psi}(k) = \mathbf{S}_{\text{D}}(k-1)\mathbf{x}(k)$

$\mathbf{S}_{\text{D}}(k) = \frac{1}{\lambda}\left[\mathbf{S}_{\text{D}}(k-1) - \frac{\boldsymbol{\psi}(k)\boldsymbol{\psi}^{\text{H}}(k)}{\lambda + \boldsymbol{\psi}^{\text{H}}(k)\mathbf{x}(k)}\right]$

$\mathbf{w}_{\text{RLS}}(k+1) = \mathbf{w}_{\text{RLS}}(k) + e^*(k)\mathbf{S}_{\text{D}}(k)\mathbf{x}(k)$

end

Problems

1.1 Warming up your mathematical skills: from Equation (1.6), compute the gradient of the LMS cost function with respect to $\mathbf{w}^*(k+1)$ and make it equal to the null vector, $\nabla_{\mathbf{w}^*(k+1)}\xi_{\text{LMS}}(k) = \mathbf{0}$, to obtain the LMS updating expression given in Equation (1.7).

1.2 In the system identification problem depicted in Figure 1.3, the linear and time-invariant unknown system has a pole in 0.8182 and a zero in −0.8182, that is, it can be represented by the difference equation

$$y(k) = x(k) + 0.8182x(k-1) + 0.8182y(k-1).$$

The input signal $x(k)$ corresponds to zero-mean white Gaussian noise (WGN) with variance $\sigma_x^2 = 1$, while the observation noise $n(k)$ is also zero-mean WGN

https://doi.org/10.1017/9781108896139.002 Published online by Cambridge University Press

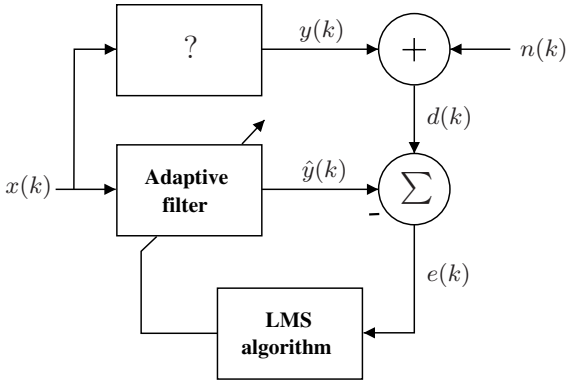

Figure 1.3 The LMS algorithm identifying an unknown system (Problem 1.2).

with variance $\sigma_n^2 = 10^{-6}$. Set the step size of the LMS algorithm to $\frac{1}{5(N+1)\sigma_x^2}$ and plot an estimate of the MSE (learning curve) with an ensemble of 100 independent runs (20 000 samples each run). Use order $N = 29$ or, equivalently, 30 coefficients. Also plot, in a dashed red line, the minimum theoretical MSE, $\xi_{\min}$, which corresponds $\mathbb{E}[e^2(k)]$ if we used $\mathbf{w}_o$, the Wiener solution. Hint: The unknown system is an infinite impulse response filter and we are using a finite impulse response adaptive filter to identify it. Therefore, we do not have perfect modeling, and $\xi_{\min}$ shall be larger than σ_n^2. See Chapter 2 in [1] for more details on how to obtain $\xi_{\min} = \sigma_n^2 + \sigma_x^2 \sum_{k=N+1}^{\infty} h^2(k)$, with $h(k)$ being the impulse response of the unknown filter, in this case.

1.3 In the same setup of the previous problem, assume that $x(k)$ corresponds to a zero-mean unitary variance WGN $r(k)$ after passed through an autoregressive (AR) system with a pole in -0.9, that is, $x(k) = r(k) - 0.9x(k-1)$. Re-run the simulation with the LMS algorithm, the NLMS algorithm, and the AP algorithm with $L = 1$ (two hyperplanes, which corresponds to the Binormalized LMS (BNLMS) algorithm). Adjust the step sizes of the NLMS and the BNLMS algorithms such that they present the same misadjustment provided by the LMS algorithm with $\mu = \frac{1}{5(N+1)\sigma_x^2}$. Note that, now that the input signal is not white, the value of $\xi_{\min}$, or its estimate, is slightly more complicated to obtain.

1.4 In forward linear prediction, we try to predict a sample of a signal $s(k)$ under analysis from its past values, that is, $d(k) = s(k)$ and $x(k) = s(k-1)$. We can use an adaptive filter to estimate the prediction coefficients $\mathbf{w} = [w_0 \ \cdots \ w_N]^{\mathrm{T}}$ as indicated in Figure 1.4(a). For generating signal $s(k)$, we assume that the optimum coefficients are given by $\mathbf{w}_o = [1.44 \ -0.68 \ -0.42 \ 0.24 \ 0.37 \ -0.35]^{\mathrm{T}}$. Signal $s(k)$ is synthesized as shown in Figure 1.4(b) where the excitation input $e(k)$ is a train of impulses with period 51 samples (meaning 6.375 ms or,

https://doi.org/10.1017/9781108896139.002 Published online by Cambridge University Press

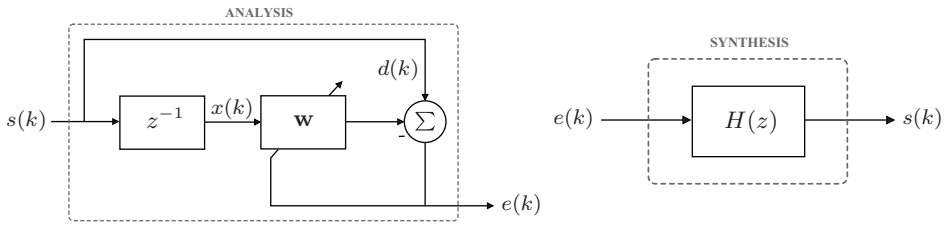

Figure 1.4 The adaptive filter in the signal prediction application.
(a) The overall filter (analysis) corresponds to $A(z) = 1 - w_0 z^{-1} \cdots - w_N z^{-(N+1)}$.
(b) Note that the synthesis filter is given as $H(z) = \frac{1}{A(z)}$.

equivalently, a "pitch" approximately equal to 156.86 Hz). The signal $s(k)$, obtained as previously described, corresponds to a segment of a synthetic vowel "ah."

Use the following commands to generate $\Delta t = 20$ ms of this voiced speech. We use Matlab® to describe the signal generation accurately but any other computer language can be easily adapted.

```
fs=8e3;           T=1/fs;
Deltat=20e-3;     t=0:T:Deltat;
exc=zeros(length(t),1);
period=51;        exc(1:period:end)=1;
wo=[1.44 -0.68 -0.42 0.24 0.37 -0.35];
A=[1 -wo];      s=filter(1,A,exc);
```

Run an adaptive filter using the LMS algorithm with step size $\mu = 0.1$, order $N = 5$, and data pair $\{d(k),x(k)\}$ as depicted in Figure 1.4. In the end, plot the optimum filter and the LMS result in the same figure. You will observe that they are quite different (the LMS did not have enough time to converge). Increase Δt and re-run the experiment; the LMS coefficient will get closer to the optimum coefficients. Further, increase Δt until the norm of the error in the coefficient vector is smaller than 0.2, that is, $\|\mathbf{w}_o - \mathbf{w}_{\text{LMS}}\| < 0.2$. How long did it take to converge? After convergence, check if the *a priori* prediction error $e(k)$ resembles the train of impulses with period 51 samples.

1.5 In Figure 1.5, we observe an adaptive signal enhancement where the interfering signal $n(k)$, after passing through an unknown room impulse response (RIR), corrupts the signal of interest (SOI) $s(k)$. In this application, the corrupted signal corresponds to the reference signal $d(k) = s(k) + n(k) * h_1(k)$. The input for an adaptive filter running the RLS algorithm is a signal also coming from the noise source but through another RIR, $x(k) = n(k) * h_2(k)$, such that it is not correlated with the SOI.

https://doi.org/10.1017/9781108896139.002 Published online by Cambridge University Press

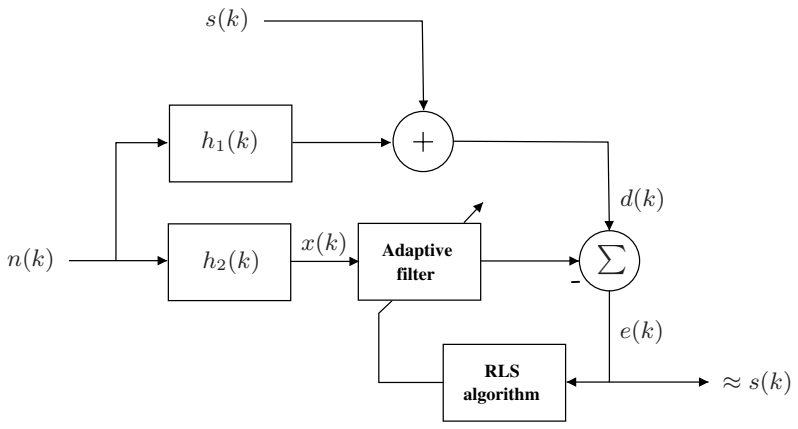

Figure 1.5 The RLS algorithm in a signal enhancement setup (Problem 1.5).

For this problem, assume that:

(a) The sampling frequency is $f_s = 8$ kHz.

(b) The SOI $s(k)$ corresponds to 4 s of speech signal (32 000 samples). Make it zero mean and unitary variance. You can record speech, use any recorded utterance, or synthesize the signal with a text-to-speech tool. If you are using Matlab® on MS® Windows operating system, you can use:
(see www.mathworks.com/matlabcentral/fileexchange/18091-text-to-speech)

```
txt='I like spring in California and winter in Rio de Janeiro.';
SV = actxserver('SAPI.SpVoice');
TK = invoke(SV,'GetVoices');
SV.Voice = TK.Item(1);
MS = actxserver('SAPI.SpMemoryStream');
MS.Format.Type = sprintf('SAFT8kHz16BitMono');
SV.AudioOutputStream = MS;
invoke(SV,'Speak',txt);
s = reshape(double(invoke(MS,'GetData')),2,[])';
s = (s(:,2)*256+s(:,1))/32768; s(s >= 1) = s(s >= 1)-2;
s=s(1:4*32000); s=s-mean(s); s=s/std(s);
```

(c) The interfering signal $n(k)$ corresponds to 4 s of the music signal. If you are using Matlab®, you can use `load handel`; make sure that the sampling frequency is the same (resample otherwise) and trim it to the same number of samples (32 000).

(d) The RIRs are $h_1(k) = \delta(k-1) + 0.5\delta(k-2)$ and $h_2(k) = \delta(k) - 0.25\delta(k-2)$.

(e) There is no observation noise.

(f) The filter order is $N = 9$ or, equivalently, the adaptive filter has 10 coefficients.

Run the RLS algorithm with $\lambda = 0.999$ and observe the $e(k)$ that should be approximately $s(k)$ after convergence. Compare the filter coefficients with $h_1(k) * h_2(k)$ which, in this case, corresponds to the optimal coefficients. Measure the SNR (in dB) of $d(k)$ (before enhancement) and of $e(k)$ (after enhancement). What is the SNR gain (in dB)?

https://doi.org/10.1017/9781108896139.002 Published online by Cambridge University Press

References

[1] P. S. R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*, 5th ed. (Cham, Switzerland, 2020).

[2] M. L. Honig and D. G. Messerschmitt, *Adaptive Filters: Structures, Algorithms, and Applications* (Kluwer Academic, Boston, 1984).

[3] S. T. Alexander, *Adaptive Signal Processing* (Springer, New York, 1986).

[4] M. Bellanger, *Adaptive Digital Filters*, 2nd ed. (Marcel Dekker, Inc., New York, 2001).

[5] B. Widrow and S. D. Stearns, *Adaptive Signal Processing* (Prentice Hall, Englewood Cliffs, 1985).

[6] J. R. Treichler, C. R. Johnson Jr., and M. G. Larimore, *Theory and Design of Adaptive Filters* (Wiley, New York, 1987).

[7] B. Farhang-Boroujeny, *Adaptive Filters: Theory and Applications* (Wiley, New York, 1998).

[8] S. Haykin, *Adaptive Filter Theory*, 4th ed. (Prentice Hall, Englewood Cliffs, 2002).

[9] A. H. Sayed, *Fundamentals of Adaptive Filtering* (Wiley, Hoboken, 2003).

[10] P. S. R. Diniz and B. Widrow, History of Adaptive Filters, in *A Short History of Circuits and Systems*, F. Maloberti and A. C. Davies (eds.) (River Publishers, Delft, 2016).

[11] B. Widrow and D. Park, History of Adaptive Signal Processing: Widrow's Group, in *A Short History of Circuits and Systems*, F. Maloberti and A. C. Davies (eds.) (River Publishers, Delft, 2016).

[12] C. F. Gauss, Theoria Combinationis Observationum Erroribus Minimis Obnoxiae: Pars Prior, Pars Posterior, Supplementum (Theory of the Combination of Observations Least Subject to Errors: Part One, Part Two, Supplement), in *Classics in Applied Mathematics*, G. W. Stewart (ed.) (SIAM, Philadelphia, 1995).

[13] J. A. Apolinário Jr. (ed.), *QRD-RLS Adaptive Filtering* (Springer, New York, 2009).

[14] F. T. Castoldi and M. L. R. de Campos, Application of a Minimum-Disturbance Description to Constrained Adaptive Filters, IEEE Signal Processing Letters **20**, pp. 1215–1218 (2013).

[15] K. Ozeki, *Theory of Affine Projection Algorithms for Adaptive Filtering* (Springer, New York, 2016).

[16] S. Werner, J. A. Apolinário Jr., and P. S. R. Diniz, Set Membership Proportionate Affine Projection Algorithms, EURASIP Journal on Audio, Speech, and Music Processing **2007**, pp. 1–10 (2007).

[17] S. Nagaraj, S. Gollamudi, S. Kapoor, and Y.-F. Huang, BEACON: An Adaptive Set-Membership Filtering Technique with Sparse Updates, IEEE Transactions on Signal Processing **47**, pp. 2928–2941 (1999).

[18] R. C. de Lamare and P. S. R. Diniz, Set-Membership Filtering Adaptive Algorithms Based on Time-Varying Error Bounds for CDMA Interference Suppression, IEEE Transactions on Vehicular Technology **58**, pp. 644–654 (2009).

[19] P. S. R. Diniz, On Data-Selective Adaptive Filtering, IEEE Transactions on Signal Processing **66**, pp. 4239–4252 (2018).

[20] F. Bouteille, P. Scalart, and M. Corazza, Pseudo Affine Projection Algorithm New Solution for Adaptive Identification. Proceedings of European Conference on Speech Communication and Technology, 1999, Vol. 1, pp. 427–430.
[21] F. Albu, M. Bouchard, and Y. Zakharov, Pseudo-Affine Projection Algorithms for Multichannel Active Noise Control, IEEE Transactions on Audio, Speech, and Language Processing **15**, pp. 1044–1052 (2007).
[22] J. Nagumo and A. Noda, A Learning Method for System Identification, IEEE Transactions on Automatic Control **12**, pp. 282–287 (1967).
[23] A. E. Albert and L. S. Gardner Jr., *Stochastic Approximation and Nonlinear Regression* (MIT Press, Cambridge, MA, 1967).
[24] B. Widrow and M. E. Hoff, Adaptive Switching Circuits, IRE WESCON Convention Record, **4**, pp. 96–104 (1960).
[25] S. Gollamudi, S. Nagaraj, S. Kapoor, and Y.-F. Huang, Set-Membership Filtering and a Set-Membership Normalized LMS Algorithm with an Adaptive Step Size, IEEE Signal Processing Letters **5**, pp. 111–114 (1998).
[26] P. S. R. Diniz and S. Werner, Set-Membership Binormalized Data Reusing LMS Algorithms, IEEE Transactions on Signal Processing **51**, pp. 124–134 (2003).
[27] S. Werner and P. S. R. Diniz, Set-Membership Affine Projection Algorithm, IEEE Signal Processing Letters **8**, pp. 231–235 (2001).
[28] M. V. S. Lima, T. N. Ferreira, W. A. Martins, and P. S. R. Diniz, Sparsity-Aware Data-Selective Adaptive Filters, IEEE Transactions on Signal Processing **62**, pp. 4557–4572 (2014).
[29] T. N. Ferreira, W. A. Martins, M. V. S. Lima, and P. S. R. Diniz, Convex Combination of Constraint Vectors for Set-Membership Affine Projection Algorithms. Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2019), Brighton, UK, 2019, pp. 4858–4862.