PART II

LINEAR ADAPTIVE ALGORITHMS

—Playing chess is about the dumbest question you can ask.
But, if you want, maybe can make money that way, or something.
Noam Chomsky

4

FUNDAMENTALS OF ADAPTIVE FILTERING

Contents

4.1 INTRODUCTION TO ADAPTIVE FILTERS

In studying *digital signal processing* (DSP) techniques, the term "adaptive" is used when a (digital or analog) system is able to automatically "adjust" its

parameters in response to input stimuli in order to achieve a processing goal [146].

An *adaptive filter* is defined as a self-designing system that relies for its operation on a recursive algorithm, which makes it possible for the filter to perform satisfactorily in an environment where knowledge of the relevant statistics is not available [59]. In that context, an adaptive filter can be viewed as an "intelligent circuit" able to adapt according to a predetermined law [146].

The ability of an adaptive filter to carry out a certain target is usually expressed through a criterion that minimizes a given *cost function*, often denoted as $J(\cdot)$, which is a function of filter parameters. The procedure which determines the variation law of the filter parameters, according to a given cost function, is also known as *adaptive algorithm*, or in same cases *learning algorithm*.

Usability of adaptive filtering techniques for the solution of real problems is widely stretched as much as fields of their applications. Adaptive filters are extensively used in many DSP areas, such as: modelling, estimate, localization, source separation, etc. Due to the rise of *neural networks*, which may be considered as a particular nonlinear class of adaptive filters, the field of interest has been further extended, thus intersecting *artificial intelligence* methods in order to provide consistent solutions even for the so-called *ill-posed* problems [146]. Recently such methods have merged into an infant subject named *computational intelligence*.

4.1.1 Classification of adaptive filters

There are a lot of way of classifying adaptive filters [146], however, the most popular classification may be carried out based on the learning algorithm and on the input-output relation.

A first subdivision concerns the adopted learning algorithm, i.e. the modality with which it is possible to adapt the filter parameters. In particular,

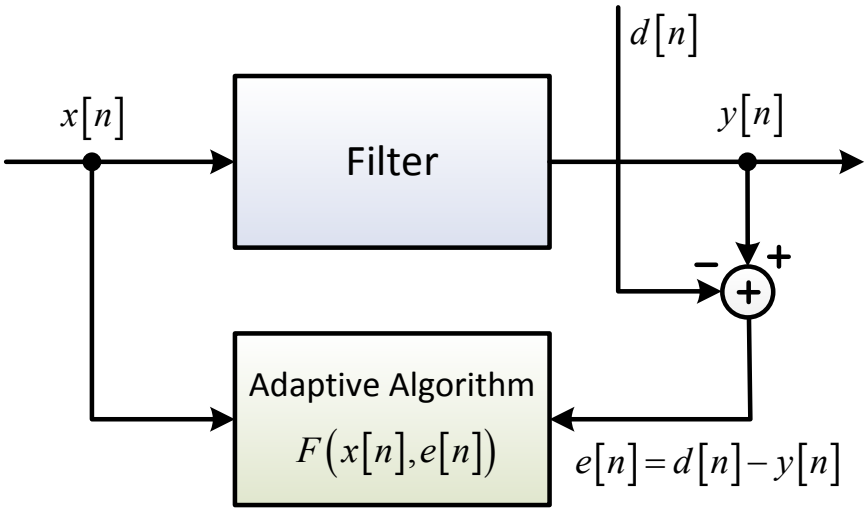

Fig. 4.1: *Scheme of a supervised adaptive filter.*

adaptive filters can be classified into:

- *supervised adaptive filters* - require the availability of a training sequence that provides different realizations of a desired response for a specified input signal vector. The desired response is compared against the actual response of the filter due to the input signal vector, and the resulting error signal is used to adjust the free parameters of the filter. The process of parameter adjustments is continued in a step-by-step fashion until a steady-state condition is established. A representation of a supervised adaptive filter is depicted in Fig. 4.1;

- *unsupervised adaptive filters* - perform adjustments of its free parameters without the need for a desired response. For the filter to perform its function, its design includes a set of rules that enable it to compute an input-output mapping with specific desirable properties. In the signal-processing literature, unsupervised adaptive filtering is often referred to as blind deconvolution or blind adaptation [59]. However, in this

dissertation we essentially deal with supervised adaptive filters.

Adaptive filters may also be classified according to an input-output relation. Denoting with $\mathbf{w}_n$ the time-varying vector of filter coefficients (i.e. filter parameters), it is possible to classify an adaptive filter according to the properties of an operator $T\{\cdot\}$ which defines the relation between the input of the filter $x[n]$ and its output $y[n]$:

$$y[n] = T\{x[n], \mathbf{w}_n\}. \tag{4.1}$$

On this basis, two main groups of adaptive filters can be characterized:

- *linear adaptive filters* - for the operator $T\{\cdot\}$ the *superposition principle* holds. Linear adaptive filters compute an estimate of a desired response by using a linear combination of the available set of observables applied to the input of the filter [59, 146];
- *nonlinear adaptive filters* - for the operator $T\{\cdot\}$ the *superposition principle* is not valid anymore [59]. In this case it is usually necessary to define further sub-labels due to the nature of the nonlinearity, that can be monodrome, invertible, uninvertible, static, dynamic, etc. [146].

Therefore, sub-labels for linear and nonlinear adaptive filters, always taking into account the input-output relation, may be the following ones:

- *static* - the output at time instant n only depends on the input at time instant n; in this case the operator $T\{\cdot\}$ has the same properties of a function;
- *dynamic with finite memory* or *FIR* - the output at time instant n depends on the input samples according to instants $n, n-1, \ldots, n-M+1$ of a time window, i.e.:

$$y[n] = T\{x[n], x[n-1], \ldots, x[n-M+1], \mathbf{w}_n\} \tag{4.2}$$

where M is the length of the time window, i.e. the *filter length*;

- *dynamic with infinite memory* or *IIR* - the output at time instant n depends on the input at time instants $n, n-1, \ldots, n-M+1$, and on past output samples, i.e.:

$$\begin{aligned} y[n] = & T\{x[n], x[n-1], \ldots, x[n-M+1], \\ & y[n-1], \ldots, y[n-M+1], \mathbf{w}_n\}. \end{aligned} \tag{4.3}$$

A possible classification of adaptive filters [146] based on the input-output relation (restricted to the dynamic case), is depicted in Fig. 4.2.

4.2 LINEAR OPTIMUM FILTERING

Linear optimum discrete-time filters are also known as *Wiener filters*, which are an extremely useful tool since its invention in the early 30's by Norbert Wiener [156]. Wiener was one of the first researchers to treat the filtering problem of estimating a process corrupted by additive noise. The optimum estimate that he derived required the solution of an integral equation known as the *Wiener-Hopf equation* [158]. Soon after he published his work, Levinson formulated the same problem in discrete time [77]. Levinson's contribution has had a great impact on the field of adaptive signal processing. Indeed, thanks to him, Wiener's ideas have become more accessible to many engineers [65]. Wiener theory plays a fundamental role in acoustic applications in which the AIR between a loudspeaker and a microphone needs to be identified. Thanks to many adaptive algorithms directly derived from the Wiener-Hopf equations, this task is now rather easy.

With the Wiener theory, it is possible to identify an unknown system, that in the acoustic case is the AIR. Given the input signal $x[n]$ and the desired signal $d[n]$ it is possible to define the error signal $e[n]$:

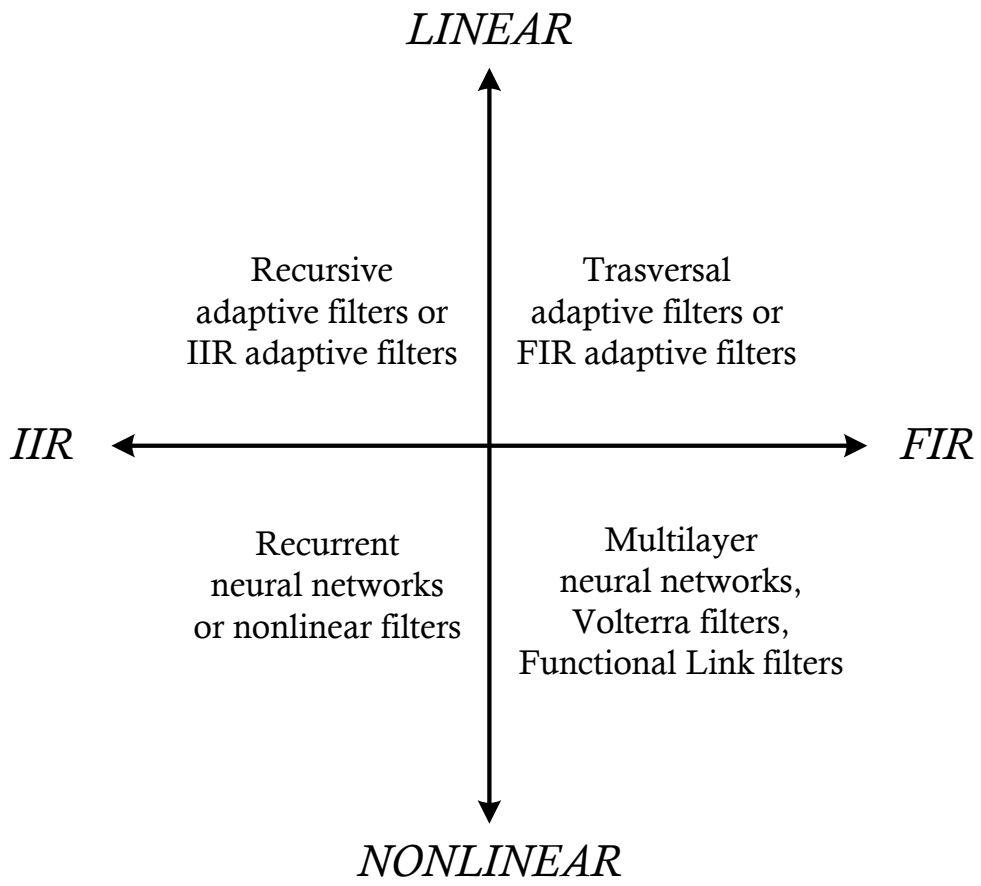

Fig. 4.2: *Classification of adaptive filters based on the input-output relation.*

$$\begin{aligned} e\left[n\right] &= d\left[n\right] - y\left[n\right] \\ &= d\left[n\right] - \mathbf{x}_n^T \mathbf{w}_{n-1} \end{aligned} \tag{4.4}$$

where $y\left[n\right]$ is the filter output and vector $\mathbf{w}_n \in \mathbb{R}^M = \left[\begin{array}{cccc} w_0\left[n\right] & w_1\left[n\right] & \dots & w_{M-1}\left[n\right] \end{array} \right]^T$ is an estimate of the AIR to identify. We suppose that the AIR and the vector $\mathbf{w}_n$ have the same length M.

To find the optimal filter, we need to minimize a cost function which is always built around the error signal (4.3) [59, 65]. The usual choice for this

criterion is the *mean square error* (MSE) [59]:

$$\begin{aligned} J(\mathbf{w}_n) =& \mathrm{E}\left\{e^2[n]\right\} \\ =& \mathrm{E}\left\{d^2[n]\right\} - \mathrm{E}\left\{\mathbf{w}_n^T\mathbf{x}_n d[n]\right\} - \mathrm{E}\left\{\mathbf{x}_n\mathbf{w}_n^T d[n]\right\} \\ & - \mathrm{E}\left\{\mathbf{w}_n^T\mathbf{x}_n\mathbf{x}_n^T\mathbf{w}_n\right\}. \end{aligned} \tag{4.5}$$

Let us remember that, for definition: $\sigma_d^2 = \mathrm{E}\left\{d^2[n]\right\}$ is the variance of the signal $d[n]$; $\mathbf{g}_n = \mathrm{E}\left\{\mathbf{x}_n d[n]\right\} \in \mathbb{R}^M$ is the cross-correlation between the input $\mathbf{x}_n$ and the desired signal $d[n]$; and, finally, $\mathbf{R}_n = \mathrm{E}\left\{\mathbf{x}_n\mathbf{x}_n^T\right\} \in \mathbb{R}^{M\times M}$ is the autocorrelation matrix. Equation (4.4) can be brought back in the following quadratic form [59, 146]:

$$J(\mathbf{w}_n) = \sigma_d^2 - \mathbf{w}_n^T\mathbf{g}_n - \mathbf{g}_n^T\mathbf{w}_n + \mathbf{w}_n^T\mathbf{R}_n\mathbf{w}_n \tag{4.6}$$

The optimal Wiener filter, that we denote as $\mathbf{w}^{\mathrm{opt}}$, is the one that cancels the gradient of $J(\mathbf{w}_n)$ with respect to $\mathbf{w}_n$, i.e.:

$$\nabla J(\mathbf{w}_n) = \frac{\partial J(\mathbf{w}_n)}{\partial \mathbf{w}_n} = \mathbf{0} \tag{4.7}$$

where the operator ∇ denotes the gradient. We have:

$$\begin{aligned} \nabla J(\mathbf{w}_n) &= 2\mathrm{E}\left\{e[n]\frac{\partial e[n]}{\partial \mathbf{w}_n}\right\} \\ &= -2\mathrm{E}\left\{e[n]\mathbf{x}_n\right\}. \end{aligned} \tag{4.8}$$

Therefore, taking into account (4.5) and (4.7), at the optimum we have:

$$\begin{aligned} \nabla J(\mathbf{w}_n) &= \frac{\partial\left(\sigma_d^2 - \mathbf{w}_n^T\mathbf{g}_n - \mathbf{g}_n^T\mathbf{w} + \mathbf{w}_n^T\mathbf{R}_n\mathbf{w}_n\right)}{\partial \mathbf{w}_n} \\ &= 2\left(\mathbf{R}_n\mathbf{w}_n - \mathbf{g}_n\right). \end{aligned} \tag{4.9}$$

Therefore, the solving system results:

$$\mathbf{R}_n \mathbf{w}_n = \mathbf{g}_n \tag{4.10}$$

which corresponds to a linear system of equations, also known as *Wiener-Hopf normal equations* [156]. The solution to (4.9), also known as *Widrow-Hopf equation* [155, 153], can be written as:

$$\mathbf{w}^{\text{opt}} = \mathbf{R}_n^{-1} \mathbf{g}_n \tag{4.11}$$

Linear optimum filtering provides minimum MSE and therefore helps to estimate accurately the unknown AIR.

4.3 GRADIENT ADAPTATION

The optimal solution to (4.9) can be obtained employing a *gradient descent* optimization procedure.

4.3.1 The steepest descent method

The method of *steepest descent gradient*, as the name implies, relies on the slope at any point on the error performance surface to provide the best direction in which to move. The steepest descent direction gives the greatest change in elevation of the surface of the cost function for a given step laterally. The steepest descent procedure uses the knowledge of this direction to move to a lower point on the surface and find the bottom of the surface in an iterative manner.

The steepest descent method is based on an iterative approach for finding the parameter value associated with the minimum of the cost function: simply move the current parameter value in the direction opposite to that of the slope of the cost function at the current parameter value. Furthermore, if we make the magnitude of the change in the parameter value proportional to the

magnitude of the slope of the cost function, the algorithm will make large adjustments of the parameter value when its value is far from the optimum value and will make smaller adjustments to the parameter value when the value is close to the optimum value [85]. This approach is the essence of the steepest descent algorithm.

The *steepest descent algorithm* can be defined considering a recursive solution to Wiener normal equations (4.10). The algorithm can be represented by its general form:

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \frac{1}{2}\mu\left(-\nabla J\left(\mathbf{w}_{n-1}\right)\right) \tag{4.12}$$

where the value $1/2$ is just a proportionality constant and the parameter μ is termed the *step size* of the algorithm. Note that for the steepest descent algorithms n is an iteration index and does not coincide with the time instant. Denoting $J\left(\mathbf{w}_n\right) = \mathrm{E}\left\{e^2\left[n\right]\right\}$, the explicit expression of the gradient $\nabla J\left(\mathbf{w}_n\right)$ can be easily derived from (4.6), thus resulting in (4.9). Therefore, replacing (4.9), evaluated at iteration index $n-1$, in (4.12), the explicit form of the steepest descent algorithm results:

$$\begin{aligned}\mathbf{w}_n &= \mathbf{w}_{n-1} - \mu\left(\mathbf{R}_{n-1}\mathbf{w}_{n-1} - \mathbf{g}_{n-1}\right)\\ &= \left(\mathbf{I} - \mu\mathbf{R}_{n-1}\right)\mathbf{w}_{n-1} + \mu\mathbf{g}_{n-1}\end{aligned} \tag{4.13}$$

where $\mathbf{I} \in \mathbb{R}^{M\times M}$ is an identity matrix (therefore it does not require any iteration index). Equation (4.13) is a recursive, multidimensional, finite different equation in the index n, with initial condition (i.c.) $\mathbf{w}_{-1}$ [146, 59].

4.3.2 Convergence of the steepest descent algorithm

Given that the stationary point of the steepest descent algorithm is the optimum *minimum mean square error* (MMSE) solution, a second, equally-important consideration is whether the algorithm converges at all. In order

to analyze the convergence properties of the steepest descent algorithm, let us consider the misalignment vector of the filter, denoted as $\mathbf{u}_n = \mathbf{w}_n - \mathbf{w}^{\text{opt}}$. Remembering (4.10), after few passages [59, 146], it results from from (4.13) that:

$$\mathbf{u}_n = (\mathbf{I} - \mu \mathbf{R}_{n-1})\, \mathbf{u}_{n-1} \tag{4.14}$$

Applying the *unitary similarity transformation* [53] on the correlation matrix $\mathbf{R}_n$, it is possible to obtain:

$$\mathbf{R}_n = \mathbf{Q}_n \mathbf{\Lambda} \mathbf{Q}_n^T = \sum_{i=0}^{M-1} \lambda_i \mathbf{q}_{n,i} \mathbf{q}_{n,i}^T \tag{4.15}$$

where $\mathbf{\Lambda} = \text{diag}\,(\lambda_0, \lambda_1, \ldots, \lambda_{M-1})$, also known as *spectral matrix*, is the diagonal matrix containing the *eigenvalues* λ_i, with $i = 0, \ldots, M-1$, of the correlation matrix $\mathbf{R}_n$. Matrix $\mathbf{Q}_n$, defined as $\mathbf{Q}_n = \begin{bmatrix} \mathbf{q}_{n,0} & \mathbf{q}_{n,1} & \ldots & \mathbf{q}_{n,M-1} \end{bmatrix}$, is known as *modal matrix* and it is composed of a set of orthogonal vectors $\mathbf{q}_{n,i}$ having unitary length, defined as *eigenvectors* of matrix $\mathbf{R}_n$. Matrix $\mathbf{Q}_n$ is orthonormal (such that $\mathbf{Q}_n^T \mathbf{Q}_n = \mathbf{I}$, i.e. $\mathbf{Q}_n^{-1} = \mathbf{Q}_n^T$).

Taking into account the decomposition (4.15), it is possible to rewrite (4.14) as:

$$\mathbf{u}_n = \left(\mathbf{I} - \mu \mathbf{Q}_{n-1} \mathbf{\Lambda} \mathbf{Q}_{n-1}^T\right) \mathbf{u}_{n-1}, \tag{4.16}$$

and setting $\widehat{\mathbf{u}}_n = \mathbf{Q}_n^T \mathbf{u}_n$, where $\widehat{\mathbf{u}}_n$ represents the rotated vector $\mathbf{u}_n$, it follows that:

$$\widehat{\mathbf{u}}_n = (\mathbf{I} - \mu \mathbf{\Lambda})\, \widehat{\mathbf{u}}_{n-1} \tag{4.17}$$

Therefore, equation (4.17) consists of a set of M decoupled difference equations of the first order, such as:

$$\widehat{u}_i\,[n] = (1 - \mu \lambda_i)\, \widehat{u}_i\,[n-1] \tag{4.18}$$

where $n > 0$ and $i = 0, \ldots, M-1$. This last equation describe all the M *natural modes* of the steepest descent algorithm. The solution to (4.18) can be determined starting from the i.c. $\widehat{u}_i[-1]$, such that, with a backward substitution, it is possible to write:

$$\widehat{u}_i[n] = (1 - \mu\lambda_i)^n \, \widehat{u}_i[n]. \tag{4.19}$$

Necessary condition so that the algorithm does not diverge, and therefore for the stability of the algorithm, is that the argument of the exponent is $|1 - \mu\lambda_i| < 1$, or, equivalently:

$$0 < \mu < \frac{2}{\lambda_i}. \tag{4.20}$$

This proves that, with an appropriate choice of the step size μ satisfying (4.20), $\widehat{u}_i[n]$ tends to zero for $n \to \infty$. This implies that:

$$\lim_{n\to\infty} \mathbf{w}_n = \mathbf{w}^{\text{opt}}, \qquad \forall \mathbf{w}_{-1} \quad (\text{i.c.}). \tag{4.21}$$

It follows that the vector $\mathbf{w}_n$ converges exponentially and exactly to the optimum.

4.4 STOCHASTIC GRADIENT ADAPTIVE ALGORITHMS

The method of steepest descent can be used to find the optimum MMSE estimate of $\mathbf{w}^{\text{opt}}$ in an iterative fashion. However, this procedure uses the statistics of the input and desired response signals and not on the actual measured signals. In practice, the input signal statistics are not known *a priori*. Moreover, if these statistics were known and if the autocorrelation matrix $\mathbf{R}_n$ was invertible, we could find the optimum solution given in (4.11) directly in one step! However, implementing this procedure exactly requires knowledge of the input signal statistics, which are almost always unknown

for real-world problems. Therefore, an approximate version of the gradient descent procedure can be applied to adjust the adaptive filter coefficients using only the measured signals [59, 120, 85, 146].

More precisely, from equations (4.12) and (4.8), it is possible to see that the steepest descent method depends on the input data and desired response signal statistics through the expectation operation that is performed on the product $e[n]\mathbf{x}_n$. This product is the gradient of the squared error function $\left(e^2[n]\right)/2$ with respect to the coefficient vector $\mathbf{w}_n$. We can consider the vector $e[n]\mathbf{x}_n$ as an approximation of the true gradient of the MSE estimation surface. This approximation is known as the *instantaneous gradient* of the MSE surface. In order to develop a useful and realizable adaptive algorithm it is possible to replace the gradient vector $\mathrm{E}\{e[n]\mathbf{x}_n\}$ in the steepest descent update in (4.8) by its instantaneous approximation $e[n]\mathbf{x}_n$. Adaptive filters that are based on the instantaneous gradient approximation are known as *stochastic gradient adaptive filters* [59, 120, 85, 146].

4.4.1 The Least Mean Square Algorithm

The *least mean square* (LMS) algorithm is the most popular memoryless stochastic gradient algorithm. Introduced by Widrow-Hoff in 1960 [153], it consists of simply considering the instantaneous squared error $e^2[n]$ instead of its expectation. The LMS algorithm can be viewed as a stochastic approximation of the steepest descent algorithm. Another important aspect concerns with the iteration index $n$of the algorithm that, in this case, coincides with the time index [146].

Denoting with $\nabla\widehat{J}(\mathbf{w}_{n-1}) \approx \nabla J(\mathbf{w}_{n-1})$ the gradient vector estimate, the general expression of the adaptation, similarly to (4.12), turns to be:

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \frac{1}{2}\mu\left(-\nabla\widehat{J}(\mathbf{w}_{n-1})\right) \tag{4.22}$$

with an *a priori error* [59, 120], or simply named *error*, defined as:

$$\begin{aligned} e[n] &= d[n] - y[n] \\ &= d[n] - \mathbf{x}_n^T \mathbf{w}_{n-1} \end{aligned} \tag{4.23}$$

The explicit expression of the gradient vector $\nabla \widehat{J}(\mathbf{w}_{n-1})$:

$$\begin{aligned} \nabla \widehat{J}(\mathbf{w}_{n-1}) &= \frac{\partial e^2[n]}{\partial \mathbf{w}_{n-1}} = 2e[n] \frac{\partial e[n]}{\partial \mathbf{w}_{n-1}} \\ &= 2e[n] \frac{\partial \left(d[n] - \mathbf{x}_n^T \mathbf{w}_{n-1}\right)}{\partial \mathbf{w}_{n-1}} = -2e[n]\mathbf{x}_n \end{aligned} \tag{4.24}$$

such that the adaptation equation (4.22) simply becomes:

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mu e[n]\mathbf{x}_n. \tag{4.25}$$

The algorithm is adjusted by the step size μ, which in this basis formulation is kept constant. Similarly to what done in the previous section for the steepest descent algorithm, it is possible to prove that the algorithm converges when:

$$0 < \mu < 2/\mu_{\max} \tag{4.26}$$

where $\lambda_{\max}$ represents the larger eigenvalue of the autocorrelation matrix of the input signal.

4.4.2 The Normalized Least Mean Square Algorithm

The *normalized least mean square* (NLMS) algorithm is structurally the same as the LMS, but it differs in the way that the filter coefficients are updated. In the LMS algorithm the weight adjustment is directly proportional to the amplitude of input vector samples according to (4.25). Therefore, when the vector $\mathbf{x}_n$ is large, the LMS suffers from a gradient noise amplification problem.

To overcome this problem, the adjustment applied to the weight vector at

each iteration is normalized with respect to the squared Euclidean norm of $\mathbf{x}_n$ [59, 120], thus the updating rule results:

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mu \frac{e\,[n]\,\mathbf{x}_n}{\delta_{\text{NLMS}} + \mathbf{x}_n^T \mathbf{x}_n} \tag{4.27}$$

with $0 < \mu \leq 2$: $\delta_{\text{NLMS}} > 0$ is the *regularization parameter* which prevents division by zero during initialization when $\mathbf{x}_n = \mathbf{0}$.

4.4.3 The Recursive Least Squares Algorithm

Least squares algorithms aim at the minimization of the sum of the squares of the difference between the desired signal and the model filter output [59, 120]. When new samples of the incoming signals are received at every iteration, the solution for the least squares problem can be computed in recursive form resulting in the *recursive least squares* (RLS) algorithms.

The RLS algorithm is known to pursue fast convergence even when the eigenvalue spread of the input signal correlation matrix is large. This algorithm has excellent performance when working in time-varying environments. All these advantages come with the cost of an increased computational complexity and some stability problems, which are not as critical in LMS-based algorithms [59, 120, 36]. The RLS can be classified as a Hessian-based algorithm, thus resulting an algorithm with *memory* [146, 42].

The cost function for this class of algorithms has the following expression:

$$\begin{aligned} \widehat{J}\,(\mathbf{w}_{n-1}) &= \sum_{i=0}^{n} \beta^{n-i} \left| e\,[n] \right|^2 \\ &= \sum_{i=0}^{n} \beta^{n-i} \left| d\,[i] - \mathbf{x}_n^T \mathbf{w}_{n-1} \right|^2 \end{aligned} \tag{4.28}$$

where the constant $0 < \beta \leq 1$, defined as *forgetting factor*, takes into account the memory of the algorithm. Therefore, the cost function depends on the actual

instantaneous error and on error samples evaluated in the past iterations with a weight continuously smaller. Note that when $\beta = 1$ the RLS consider the same weight for all the past samples; in that case the algorithm has a *growing memory*.

Let us take into account the following *sequential regression notation* with an input data matrix $\mathbf{X}_n \in \mathbb{R}^{N \times M}$, where N is the length of the analysis window, defined as:

$$\mathbf{X}_n = \begin{bmatrix} \mathbf{x}_n^T \\ \mathbf{x}_{n-1}^T \\ \cdots \\ \mathbf{x}_{n-N+1}^T \end{bmatrix}^T \tag{4.29}$$

$$= \begin{bmatrix} x[n] & x[n-1] & \cdots & x[n-M+1] \\ x[n-1] & x[n-2] & \cdots & x[n-M] \\ \vdots & \vdots & \ddots & \vdots \\ x[n-N+1] & x[n-N] & \cdots & x[n-N-M+2] \end{bmatrix}$$

As a consequence the error vector and the desired signal vector are respectively defined as:

$$\begin{aligned} \mathbf{e}_n \in \mathbb{R}^N &= \begin{bmatrix} e[n] & e[n-1] & e[n-N+1] \end{bmatrix} \\ \mathbf{d}_n \in \mathbb{R}^N &= \begin{bmatrix} d[n] & d[n-1] & d[n-N+1] \end{bmatrix} \end{aligned} \tag{4.30}$$

Therefore, equation (4.28) can be expressed in the regression notation [146] as:

$$\widehat{J}(\mathbf{w}_{n-1}) = \mathbf{e}_n^T \mathbf{B}_n \mathbf{e}_n = \mathbf{B}_n \left\| \mathbf{d}_n - \mathbf{X}_n \mathbf{w}_{n-1} \right\|_2^2 . \tag{4.31}$$

where $\mathbf{B}_n$ represents a weighted matrix:

$$\mathbf{B}_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & \cdots & 0 \\ \vdots & \vdots & \beta^{n-1} & \vdots \\ 0 & 0 & \cdots & \beta^n \end{bmatrix}. \tag{4.32}$$

Solving for the cost function (4.31), after few passages [59, 120, 146], it is possible to achieve the following *regression equation*:

$$\mathbf{X}_n^T \mathbf{B}_n \mathbf{X}_n \mathbf{w}_{n-1} = \mathbf{X}_n^T \mathbf{B}_n \mathbf{d}_n \tag{4.33}$$

Denoting the correlation estimates as:

$$\mathbf{R}_{xx,n} = \mathbf{X}_n^T \mathbf{B}_n \mathbf{X}_n \mathbf{w}_{n-1} \quad \text{and} \quad \mathbf{R}_{xd,n} = \mathbf{X}_n^T \mathbf{B}_n \mathbf{d}_n \tag{4.34}$$

that can be also expressed as:

$$\mathbf{R}_{xx,n} = \sum_{i=0}^{n} \beta^{n-1} \mathbf{x}_i \mathbf{x}_i^T = \beta \mathbf{R}_{xx,n-1} + \mathbf{x}_i \mathbf{x}_i^T \tag{4.35}$$

$$\mathbf{R}_{xd,n} = \sum_{i=0}^{n} \beta^{n-1} \mathbf{x}_i d\,[i] = \beta \mathbf{R}_{xd,n-1} + \mathbf{x}_i d\,[i] \tag{4.36}$$

such that the correlations can be computed in a recursive way updating the estimate carried out at the past iteration with new available information. The solution of the sequential regression (4.33) at n-th time instant can be written as:

$$\mathbf{R}_{xx,n} \mathbf{w}_{n-1} = \mathbf{R}_{xd,n} \tag{4.37}$$

Applying the *matrix inversion lemma* [53, 59, 146] to the matrix (4.35) and setting $\mathbf{P}_n = \mathbf{R}_{xx,n}^{-1}$, it is possible to achieve:

$$\mathbf{P}_n = \beta^{-1} \mathbf{P}_{n-1} - \frac{\beta^{-1} \mathbf{P}_{n-1} \mathbf{x}_n \beta^{-1} \mathbf{x}_n^T \mathbf{P}_{n-1}}{1 + \beta^{-1} \mathbf{x}_n^T \mathbf{P}_{n-1} \mathbf{x}_n} \tag{4.38}$$

where for computational convenience it is usual to define the vector:

$$\mathbf{k}_n = \frac{\beta^{-1}\mathbf{P}_{n-1}\mathbf{x}_n}{1 + \beta^{-1}\mathbf{x}_n^T\mathbf{P}_{n-1}\mathbf{x}_n} \tag{4.39}$$

also known as *Kalman gain*, so that the recursion (4.38) can be written as:

$$\mathbf{P}_n = \beta^{-1}\mathbf{P}_{n-1} - \beta^{-1}\mathbf{k}_n\mathbf{x}_n^T\mathbf{P}_{n-1} \tag{4.40}$$

also known as *Riccati equation*.

The main drawback of the RLS algorithm is its computational cost, thus LMS based algorithms, while they do not perform as well as RLS, are more favourable in practical situations.

4.4.4 The Affine Projection Algorithm

The *affine projection algorithm* (APA) can be interpreted as a generalization of the NLMS algorithm. The main advantage of the APA over the NLMS algorithm consists of a superior convergence rate, especially for correlated inputs, like speech. For this reason, the APA and different versions of it were found to be very attractive choices for acoustic applications, such as AEC.

The APA, originally proposed in [98], was derived as a generalization of the NLMS algorithm, in the sense that a filter vector of the NLMS may be viewed as a one dimensional affine projection, while in the APA the projections are made in multiple dimensions. When the projection dimension increases, the convergence rate of the filter vector also increases. However, this also leads to an increased computational complexity. The APA, like the RLS is a Hessian-based algorithm, however it is not an "exact" second order adaptive algorithm since its adaptation uses an estimate of the correlation matrix $\mathbf{R}_{xx,n}$ "projected" over a subspace with appropriate dimension [146].

In order to derive the classical APA equations, let us consider an FIR adaptive filter of length M, defined by the coefficients vector $\mathbf{w}_n$, and an input data matrix defined similarly to (4.29) but using a window length N equal

to $K > 0$, which is also defined as *projection order*. Therefore, the input data matrix is defined as $\mathbf{X}_n \in \mathbb{R}^{K \times M}$, while the error signal and the desired signal are respectively $\mathbf{e}_n$, $\mathbf{d}_n \in \mathbf{R}^K$, similarly to (4.30). This corresponds to take into account the last K samples of the input sequence. When $K = 1$ the adaptation becomes one dimensional and thus the APA turns to be an NLMS algorithm. Therefore, the equations that define the classical APA are [98]:

$$\mathbf{e}_n = \mathbf{d}_n - \mathbf{X}_n \mathbf{w}_{n-1} \tag{4.41}$$

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mu \mathbf{X}_n^T \left(\delta_{\text{APA}} \mathbf{I} + \mathbf{X}_n \mathbf{X}_n^T \right)^{-1} \mathbf{e}_n \tag{4.42}$$

where δ_{APA} is the *regularization factor* of the APA and $\mathbf{I} \in \mathbb{R}^{K \times K}$ is an identity matrix.

We will see in the next chapter a general framework for the derivation of adaptive algorithms, both for these classical stochastic gradient algorithms and for the proportionate algorithm that will be introduced in the next chapter.