2 Adaptive Filtering for Sparse Models

2.1 Introduction

In adaptive filtering, as well as in other areas requiring learning from data, there is a current trend of employing complex models relying on a large number of coefficients/parameters, meaning that the parameters form a vector, called *coefficient vector* or *parameter vector*, that lives in a high-dimensional space. In general, as the parameter space enlarges (has more dimensions), the training process becomes more difficult, and it is expected that the coefficient vector takes more time to converge to an acceptable estimate, that is, a longer training period is required. However, slow convergence is a critical issue in adaptive filtering, as its most interesting and practical applications very often deal with the tracking of nonstationary processes, thus requiring fast adaptation of the filter coefficients.

One way of overcoming this slow convergence issue is by exploiting some structure that the coefficients should have. The idea is adding prior knowledge about the problem so that the algorithm can explore the parameter space (or search space) wisely, like if the algorithm had a clue of where to search for the solution, instead of performing a simple uninformed search throughout the parameter space. Clearly, the impact on convergence speed provided by exploiting the parameters' structure increases with the number of parameters, that is, with the dimension of the search space.

There are many different parameter structures that can be exploited; some examples can be found in [1–6]. But one of the most important structures, which is the subject of this chapter, is the *sparsity*. Intuitively, a sparse parameter vector means that only a subset of the inputs is required to explain/model the data (later we will define the concepts of sparse and compressible vectors precisely) and, therefore, is very likely to happen when using complex models with a considerable amount of coefficients.

Exploiting sparsity in adaptive filtering has been a very active research topic since the early twenty-first century, as the classical algorithms have become inappropriate to deal with such a large set of coefficients. The goal of this chapter is to equip the reader with the fundamentals of this field. That is, instead of covering a huge number of adaptive filtering algorithms that exploit sparsity (there are hundreds of them), we opted for a unified presentation in which we

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

divide these algorithms into two classes. We present each class's fundamental ideas, discuss the pros and cons, and introduce the most important or pioneer algorithms of each class.

This chapter is organized as follows. In Section 2.2, we provide a thorough explanation of the sparsity-promoting functions frequently used as regularizations, the so-called *sparsity-promoting regularizations*. This section is of paramount importance to fully understand the *algorithms employing sparsity-promoting regularizations*, which is an important class of algorithms that exploit sparsity, covered in Section 2.3. In Section 2.4, we present the *proportionate-type* algorithms, another class of algorithms widely used to exploit sparsity. The conclusions are drawn in Section 2.5. We hope the reader can appreciate the art of developing algorithms during the exposition of these two classes of algorithms. That is, while the development of the algorithms employing sparsity-promoting regularizations relies on optimization theory and geometric aspects, the development of the proportionate-type algorithms relies more on numerical analysis and intuition/heuristics.

2.2 Exploiting Sparsity through Regularization

Regularization has been widely used in areas like statistics, machine learning, and adaptive filtering. *Regularization can be regarded as a way of adding prior information about the model/parameters to be estimated or of encouraging them to have some structure.* Of course, adding a regularization changes the original (nonregularized) optimization problem and, therefore, the solutions of these two distinct problems are not necessarily equal. However, the discrepancy between these two solutions can be controlled by the regularization parameter, a positive real number that determines the weight given to the regularization term.

There are many benefits of using regularization, among which we highlight: *preventing the overfitting problem* of the parameters and *enhancing the problem conditioning*. Preventing the overfitting problem is crucial for the learning technique to perform well using new data, that is, data not belonging to the training set; this characteristic is known in machine learning as *generalization* [7]. The conditioning, or condition number, is related to the problem (and not the algorithm employed to solve it) and determines how the errors/uncertainties in the observations are propagated to the unknowns/estimates [8]. If the problem is ill-conditioned (i.e., has large condition number), then these errors can be severely amplified to the outputs no matter the algorithm that is used to solve it. In this case, it is better to make a slight modification to the original optimization problem, in order to enhance its conditioning, before solving it. Besides, the condition number also impacts the learning rate of gradient-based algorithms [9].

In addition to the aforementioned benefits, by employing regularization it might be possible to obtain estimators with reduced mean squared error (MSE)

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

in comparison with unbiased solutions. As explained in [10], these estimators can "trade a little bias for a large reduction in variance."[1]

From the optimization point of view, the regularization term is simply a penalty function of the parameters. In adaptive filtering, in most cases, this penalty function is a vector norm, for example, the ℓ_2 norm of the coefficients $\mathbf{w}$. In this section, we are interested in the regularizations used to exploit sparsity, frequently called *sparsity-promoting regularizations* [11].

Recalling that a finite-dimensional vector, herein also called *signal*, is said to be sparse if it can be represented as a linear combination of a small number of basis vectors for some basis of the related vector space [12]. For instance, the vector $[0 \ \ldots \ 0 \ 1 \ 0 \ldots \ 0]^{\mathrm{T}}$ is sparse as it requires a single basis vector for its representation; the nonzero entry being equal to 1 or $1,000$ does not change this fact. Therefore, it is natural to think of the ℓ_0 norm, which essentially counts the number of nonzero entries in a vector, when modeling or measuring the sparsity of a signal. However, minimizing this norm is challenging since it leads to an NP-hard problem, which turns its use prohibitive in many cases, especially in online applications [12]. Due to the practical limitations of using the ℓ_0 norm, several alternatives have been proposed, like the ℓ_1 norm, the reweighted ℓ_1 norm, and the approximation of the ℓ_0 norm [11]. In this section, all of these approaches to model sparsity are described, their gradient expressions are shown (as gradient-based optimization is widely used), and their pros and cons are discussed.

2.2.1 The ℓ_0 Norm

As previously explained, the ℓ_0 norm is the natural function to model/measure the sparsity of a vector. So, why searching for alternatives to the ℓ_0 norm instead of working directly with it? Here, we answer this question by pointing out the practical limitations of this norm.

The ℓ_0 norm of a given vector $\mathbf{w} = [w_0 \ w_1 \ \ldots \ w_N]^{\mathrm{T}} \in \mathbb{R}^{N+1}$ is defined as

$$\|\mathbf{w}\|_0 \triangleq \#\{i \in \mathcal{I} : w_i \neq 0\} \in \mathbb{N}, \tag{2.1}$$

where $\mathcal{I} \triangleq \{0, 1, \ldots, N\} \subset \mathbb{N}$ is the set of indexes for the entries of $\mathbf{w}$ and $\#$ denotes the cardinality (i.e., the number of elements) of a finite set. Thus, the ℓ_0 norm of a vector is the number of its nonzero entries. Observe that the ℓ_0 norm is not really a norm as the homogeneity property fails, that is, $\|\alpha\mathbf{w}\|_0 \neq |\alpha|\|\mathbf{w}\|_0$ in general.[2]

A major issue related to the ℓ_0 norm concerns the *computational complexity* involving its minimization. Indeed, the problem of minimizing $\|\mathbf{w}\|_0$ subject to some constraints on $\mathbf{w}$ is said to be NP-hard and, in general, requires a combinatorial search over all possible coordinate combinations; there are $(N+1)$ choose

[1] Recall that (MSE) = (bias)^2 + (variance).

[2] Also, the ℓ_0 norm is not a pseudo-norm nor a quasi-norm, as sometimes stated erroneously, since both of these concepts relax other properties of a norm, but not homogeneity [13, 14].

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

k possible combinations, for each $k \in \{1, 2, \ldots, N+1\}$. Clearly, a tough problem with a prohibitive complexity for vectors belonging to high-dimensional spaces or when dealing with online and real-time processing requirements.

Still considering the optimization problem, but from another perspective, the use of the ℓ_0 norm leads to an *ill-conditioned problem* since the ℓ_0 norm is very sensitive to nonzero coordinates due to its discontinuity. For example, consider the following two vectors belonging to $\mathbb{R}^{N+1}$

$$\begin{aligned} \mathbf{0} &= [0 \ \ldots \ 0]^{\mathrm{T}} \qquad \text{(the vector of zeros)}, \\ \mathbf{1} &= [1 \ \ldots \ 1]^{\mathrm{T}} \qquad \text{(the vector of ones)}. \end{aligned}$$

Clearly, $\|\mathbf{0}\|_0 = 0$. If you add a tiny perturbation to this vector, the result changes by a significant amount, that is,

$$\|\mathbf{0} + \delta\mathbf{1}\|_0 = N + 1,$$

for any $\delta \neq 0$. This characteristic hinders the use of the ℓ_0 norm in most practical applications since typically there are several sources of errors/uncertainties due to noise, quantization, numerical approximations, limited accuracy of the measurement devices, etc.

From the discussion in the previous paragraph, one may conclude that sparse signals (in the mathematical sense) are not frequently found in practice.[3] Actually, many times in the literature, the term sparse is used with a slightly different meaning; it is used for vectors having their energy concentrated mostly on a few entries, whereas their remaining elements have small or negligible energy. This kind of vector/signal is herein called *compressible* [12]. Roughly, the main difference between sparse and *compressible signals* is that the negligible coefficients of the latter are not required to be 0.

Even though there is a strong relationship between sparsity and the ℓ_0 norm, its practical use is very limited due to the severe drawbacks presented here. Next, we present the classical alternatives to the ℓ_0 norm.

2.2.2 The ℓ_1 Norm

Undoubtedly, the ℓ_1 norm is the most widely used substitute for the ℓ_0 norm. It has been used in areas like geophysics since the 1970s [15, 16], but it was in the 1990s that it became widely used due to the advent of techniques like the basis pursuit [17] and the least absolute shrinkage and selection operator (LASSO) [10, 18]. Since 2006, the ℓ_1 norm has also been used to recover sparse signals, the so-called compressed sensing (CS) [19]. The CS theory explained that under certain conditions, the ℓ_1 minimization is capable of finding the sparsest solution (i.e., the solution to the ℓ_0 minimization).

[3] Sparse signals are usually obtained by replacing the small coefficients of a real signal (like an image, audio, or the impulse response of a system) with zeros. That is, the real and compressible signal goes through a lossy compression process to generate a sparse signal.

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

The ℓ_1 norm of a given signal $\mathbf{w} \in \mathbb{R}^{N+1}$ is defined as

$$\|\mathbf{w}\|_1 \triangleq \sum_{i=0}^{N} |w_i|. \tag{2.2}$$

The ℓ_1 norm is, therefore, a continuous and convex function, meaning that there exist several efficient methods to solve the ℓ_1 minimization problem, and it is almost everywhere (a.e.) differentiable, thus turning gradient-based methods very suitable to its minimization. This simplicity and mathematical tractability have played a central role in the widespread use of the ℓ_1 norm.

The derivative of $\|\mathbf{w}\|_1$ with respect to a given entry w_i is[4]

$$\frac{\partial \|\mathbf{w}\|_1}{\partial w_i} = \text{sign}(w_i) \triangleq \begin{cases} 1 & \text{if } w_i > 0, \\ 0 & \text{if } w_i = 0, \\ -1 & \text{if } w_i < 0, \end{cases} \tag{2.3}$$

that is, the standard sign function applied to a real scalar. Thus, the gradient of $\|\mathbf{w}\|_1$ with respect to $\mathbf{w}$, denoted by $\nabla\|\mathbf{w}\|_1$ or $\mathbf{f}_{\ell_1}(\mathbf{w})$, is

$$\nabla\|\mathbf{w}\|_1 \triangleq \mathbf{f}_{\ell_1}(\mathbf{w}) = \text{sign}(\mathbf{w}) \triangleq [\text{sign}(w_0)\ \text{sign}(w_1)\ \ldots\ \text{sign}(w_N)]^{\mathrm{T}}, \tag{2.4}$$

that is, when the operator sign is applied to a vector, it results in a vector with the standard sign function applied to each of its entries.

Despite the aforementioned advantages, the ℓ_1 norm has some critical drawbacks that motivated the search for other functions to exploit sparsity. These main drawbacks are as follows [11, 20–22]:

1 The ℓ_1 norm penalizes the larger coefficients more heavily than the smaller ones.
2 For gradient-based methods, the update term given by $-\nabla\|\mathbf{w}\|_1$ pushes all entries of $\mathbf{w}$ toward 0 with the same strength, instead of prioritizing those which are closer to 0.
3 The effectiveness of the ℓ_1 norm depends on the existence of high sparsity degree.

Observe that a good sparsity-promoting function should act in the small magnitude coefficients, pushing them toward 0 to encourage sparse solutions, without interfering in the large magnitude coefficients, which are relevant to explain/model the data [11]. The need to discriminate between low and high magnitude entries in order to force the former ones to 0 without shrinking the latter ones paved the way for further developments.

[4] Formally, $\|\mathbf{w}\|_1$ is not differentiable at the points $w_i = 0$; therefore, one must use the concept of subderivative at these points. For instance, the subdifferential of the function $|w|$ at $w = 0$ is the interval $[-1, 1]$, and any point $w_0 \in [-1, 1]$ is called a subderivative of $|w|$ at $w = 0$.

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

2.2.3 The Reweighted ℓ_1 Norm

The reweighted ℓ_1 ($r\ell_1$) norm proposed in 2008 aims to mitigate the problem of shrinking the large magnitude coefficients [20]. To do so, instead of penalizing each coefficient based on its magnitude $|w_i|$, as the ℓ_1 norm does, the $r\ell_1$ norm uses a function that grows less quickly for the large magnitude coefficients.

The $r\ell_1$ norm of a given signal $\mathbf{w} \in \mathbb{R}^{N+1}$, denoted by $F_{r\ell_1} : \mathbb{R}^{N+1} \longrightarrow \mathbb{R}_+$, is defined as

$$F_{r\ell_1}(\mathbf{w}) \triangleq \frac{1}{\log(1+\epsilon)} \sum_{i=0}^{N} \log\left(1+\epsilon|w_i|\right), \tag{2.5}$$

where $\epsilon \in \mathbb{R}_+$. Like the ℓ_0 norm, the $r\ell_1$ norm also is not really a norm. To better understand the role of ϵ, observe that $F_{r\ell_1}$ tends to the ℓ_1 norm as $\epsilon \to 0$, whereas $F_{r\ell_1}$ tends to the ℓ_0 norm as $\epsilon \to \infty$. The $r\ell_1$ norm is a continuous and a.e. differentiable function, but it is not convex anymore due to the logarithm. Nonetheless, gradient-based methods can still be used in its minimization [20].

The derivative of $F_{r\ell_1}(\mathbf{w})$ with respect to a given entry w_i is

$$\frac{\partial F_{r\ell_1}(\mathbf{w})}{\partial w_i} = \frac{\epsilon}{\log(1+\epsilon)} \times \frac{\text{sign}(w_i)}{1+\epsilon|w_i|}. \tag{2.6}$$

Notice that the first fraction on the right-hand side of Equation (2.6) does not depend on i and, therefore, it only modifies the length of the gradient vector, not its direction. Moreover, there already exists a parameter, *the regularization parameter*, which controls the weight given to $F_{r\ell_1}$, and thus the length of its gradient. Therefore, it is very common to discard this fraction and write the gradient vector as [19, 23]:

$$\nabla F_{r\ell_1}(\mathbf{w}) \triangleq \mathbf{f}_{r\ell_1}(\mathbf{w}) \triangleq \left[f_{r\ell_1}(w_0)\ f_{r\ell_1}(w_1)\ \ldots\ f_{r\ell_1}(w_N)\right]^{\mathrm{T}}, \tag{2.7}$$

where $f_{r\ell_1}(w_i)$ is a simplified version of Equation (2.6) given by

$$f_{r\ell_1}(w_i) = \frac{\text{sign}(w_i)}{1+\epsilon|w_i|}. \tag{2.8}$$

The $r\ell_1$ norm given in Equation (2.5), therefore, mitigates the problem of heavy penalization of the large magnitude coefficients. Besides, observe in Equation (2.8) that $f_{r\ell_1}(w_i)$ decreases as $|w_i|$ increases, meaning that gradient-based algorithms employing the $r\ell_1$ norm attract the smaller coefficients to 0 more strongly than the large magnitude ones. However, it still affects/shrinks the large magnitude coefficients, a problem that is solved by the function presented in Subsection 2.2.4.

2.2.4 The ℓ_0-Norm Approximation

The ℓ_0-*norm approximation* is also known as *smoothed* ℓ_0 *norm* because it is a continuous/smoothed version of the ℓ_0 norm. Although the functions commonly used as ℓ_0-norm approximations are known for a long time, only recently they

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

have been used for the purpose of exploiting sparsity. For instance, the ℓ_0-norm approximation has been used in image processing [22], in medical image reconstruction [24], and in system identification [11, 21, 25]. Most of these works present results pointing out the superior performance of the ℓ_0-norm approximation over the ℓ_1 norm and the $r\ell_1$ norm.

The ℓ_0-norm approximation is a continuous and a.e. differentiable function $F_\beta : \mathbb{R}^{N+1} \longrightarrow [0, N+1] \subset \mathbb{R}_+$, in which $\beta \in \mathbb{R}_+$ is a parameter that controls the smoothness of the approximation. A common practice is to define analytically a continuous function F_β such that

$$\lim_{\beta \longrightarrow \infty} F_\beta(\mathbf{w}) = \|\mathbf{w}\|_0, \tag{2.9}$$

or equivalently,

$$\lim_{\beta \longrightarrow \infty} F_\beta(w_i \mathbf{e}_i) = \begin{cases} 1 & \text{if } w_i \neq 0, \\ 0 & \text{if } w_i = 0, \end{cases} \tag{2.10}$$

for all $\mathbf{w} \in \mathbb{R}^{N+1}$, where $\mathbf{e}_i$ is the i-th vector of the canonical basis of $\mathbb{R}^{N+1}$ (i.e., $\mathbf{e}_i$ has only zero elements, except for a 1 in its i-th coordinate, $i \in \mathcal{I}$). For finite values of β, we add the following property:

$$F_\beta(w_i \mathbf{e}_i) = \begin{cases} 1 & \text{if } |w_i| \gg 1/\beta, \\ 0 & \text{if } w_i = 0. \end{cases} \tag{2.11}$$

In words, a valid ℓ_0-norm approximation must be a continuous function whose value is equal to 0 at $w_i = 0$ and rapidly (depending on the value of β) saturates at 1 as w_i gets further from 0, for every coordinate w_i. Clearly, F_β is nonconvex. It is worth noticing that the reweighted ℓ_1 norm, denoted by $F_{r\ell_1}$, also converges to the ℓ_0 norm as $\epsilon \to \infty$. However, for finite ϵ, $F_{r\ell_1}$ increases with $|w_i|$, thus not satisfying the saturation property given in Equation (2.11). Therefore, $F_{r\ell_1}$ is not a particular case of F_β.

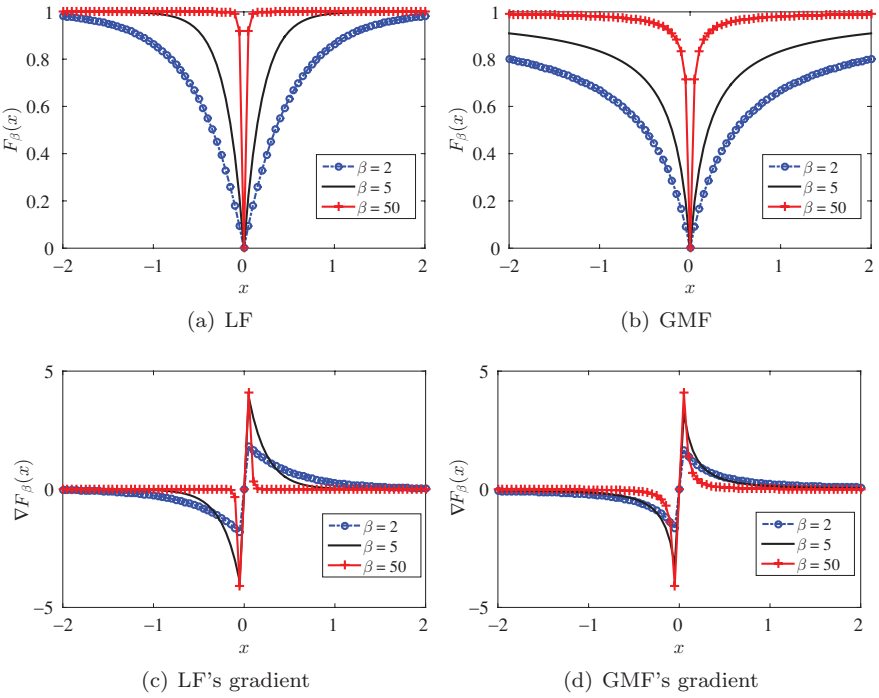
There are many functions F_β that can be used as ℓ_0-norm approximations, as illustrated in [11]. Here, we focus on the two most commonly used:

$$F_\beta(\mathbf{w}) = \sum_{i \in \mathcal{I}} \left(1 - \mathrm{e}^{-\beta |w_i|}\right), \tag{2.12a}$$

$$F_\beta(\mathbf{w}) = \sum_{i \in \mathcal{I}} \left(1 - \frac{1}{1 + \beta |w_i|}\right). \tag{2.12b}$$

The F_β given in Equation (2.12a) is the so-called Laplace function (LF) [24, 26], and is probably the most widely used ℓ_0-norm approximation. In addition, Equation (2.12b) describes the Geman–McClure function (GMF) [24, 27]. Notice that both approximations satisfy the properties in Equations (2.10) and (2.11).

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press



(a) LF (b) GMF

(c) LF's gradient (d) GMF's gradient

Figure 2.1 Functions $F_\beta(x)$ and their gradients $\nabla F_\beta(x)$, with $x \in \mathbb{R}$, for different values of β: (a) Laplace function (LF); (b) Geman–McClure function (GMF); (c) LF's gradient; and (d) GMF's gradient.

Defining $f_\beta(w_i) \triangleq \frac{\partial F_\beta(\mathbf{w})}{\partial w_i}$, the derivatives corresponding to Equation (2.12a) and Equation (2.12b) are, respectively,

$$f_\beta(w_i) = \beta \text{sign}(w_i) \mathrm{e}^{-\beta|w_i|}, \tag{2.13a}$$

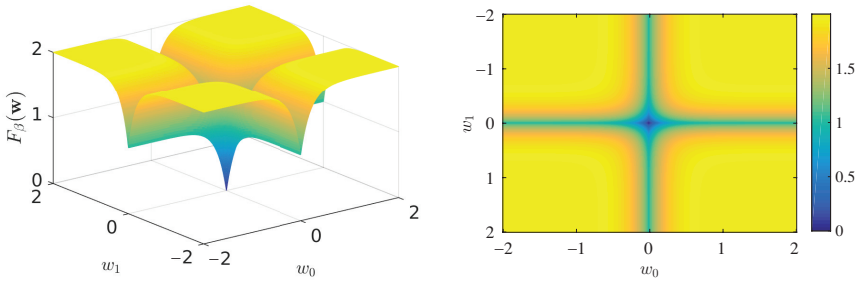
$$f_\beta(w_i) = \frac{\beta \text{sign}(w_i)}{(1+\beta|w_i|)^2}, \tag{2.13b}$$

where the function $\text{sign} : \mathbb{R} \longrightarrow \{-1, 0, 1\}$ is defined in Equation (2.3). Thus, we can define the gradient of $F_\beta(\mathbf{w})$ with respect to $\mathbf{w}$ as

$$\nabla F_\beta(\mathbf{w}) \triangleq \mathbf{f}_\beta(\mathbf{w}) \triangleq [f_\beta(w_0)\ f_\beta(w_1)\ \ldots\ f_\beta(w_N)]^{\mathrm{T}}. \tag{2.14}$$

Figure 2.1 depicts the LF, the GMF, and their respective gradients $\nabla F_\beta(x) = \mathrm{d}F_\beta(x)/\mathrm{d}x$, for $x \in \mathbb{R}$, using different values of β. Observe that the functions F_β are nonconvex and also that low values of β lead to smoother functions, whereas high values of β turn these functions more similar to the ℓ_0 norm. Besides, due to the exponential function, the LF goes from 0 to 1 faster than the GMF, at least for the same value of β, thus being a better approximation of the ℓ_0 norm. On the other hand, the GMF is smoother and simpler to compute. Also, observe that the gradients of both the LF and GMF are very strong (high

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press



(a) LF surface (b) Top view of the LF surface

Figure 2.2 Laplace function F_β for $\beta = 5$ and $\mathbf{w} \in \mathbb{R}^2$: (a) LF surface and (b) top view of the LF surface.

magnitude) at values of x close to 0 and they vanish (i.e., $\nabla F_\beta(x) = 0$) as $|x|$ grows. Clearly, the gradient of the LF vanishes faster than that of the GMF, for a fixed β.

Figure 2.2 illustrates the surface of the LF $F_\beta(\mathbf{w})$, with $\mathbf{w} \in \mathbb{R}^2$ and $\beta = 5$. Observe that such function has a single global minimum at $\mathbf{w} = \mathbf{0}$ and there are infinitely many local minima at the axes, more precisely, when one of the coordinates is equal to 0, and the other one is sufficiently large. It is worth mentioning that most of the parameter space is covered by vast plateaus, whereas the concave part of the function occurs when any of the coefficients w_i approaches 0, as can be clearly observed in Figure 2.2(b).

From the previous discussion, it must be clear that the ℓ_0-norm approximation F_β does not have the drawbacks of the previous functions. Indeed, due to its saturation property, the high magnitude (relevant) coefficients are not heavily penalized, and a gradient-based minimization pushes only the small magnitude coefficients to 0, keeping the relevant coefficients unaltered. Also, the ℓ_0-norm approximation works well even when the sparsity degree is very low, as corroborated by the results and theoretical analysis in [28].

Gradient-Based Training

As illustrated in Figure 2.1, the parameter β represents a tradeoff between smoothness and quality of approximation. Therefore, this parameter can be used to avoid gradient-descent techniques from getting trapped in a local minimum of F_β, simply by reducing the value of β in order to obtain a smoother approximation. For instance, by decreasing the value of β in Figure 2.2, one can obtain a LF surface that does not exhibit local minima within the $[-2, 2] \times [-2, 2]$ region and, therefore, gradient-based techniques will work well within this region.[5] The price to be paid is to have a slower convergence due to the

[5] F_β will always have local minima, but as you set β closer to 0, these local minima occur much further from the origin. That is, one can always set β so that the local minima appear in a part of the parameter space that is of no interest.

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

smaller gradients since, for a given x close to 0, the magnitudes of the gradients decrease as β decreases, as can be observed in Figure 2.1.

To overcome the aforementioned issue, a fast algorithm for overcomplete sparse decomposition, called SL0, was proposed in [29]. Essentially, the SL0 algorithm uses an increasing sequence $\beta_1 < \beta_2 < \cdots < \beta_J$, applies the steepest-descent technique (or steepest-ascent, if the problem is written in the form of maximization, as in [29]) to solve approximately the minimization of F_{β_i}, and then uses the approximate solution of the i-th problem as the initialization of the next problem $F_{\beta_{i+1}}$. The process is repeated until F_{β_J} is solved. It is assumed that β_J is sufficiently large so that F_{β_J} is a good approximation of the ℓ_0 norm. The work in [29] also provides some theoretical guarantees that the SL0 solution is indeed the sparsest solution, that is, the one that minimizes the ℓ_0 norm. However, the choice of the sequence of β's is addressed only heuristically.

The same idea of the SL0 algorithm could be adapted for adaptive filtering by employing an iteration-dependent parameter $\beta(k)$, where $k \in \mathbb{Z}$ is the iteration index and defining some adaptation rule for it. Indeed, unlike in sparse decomposition applications, in which the SL0 is satisfactorily used, working with a predefined and always increasing sequence of β's is usually not possible in adaptive filtering since very often the problem involves nonstationary signals. In this way, a recursion capable of increasing and decreasing the value of $\beta(k)$ would be necessary, a topic that, to the best of our knowledge, has never been addressed. In what follows, we explain how β is usually chosen.

Choice of β

For the proper choice of β, we must introduce the concept of *zero-attraction region*, which is a region of the parameter space where the gradient is non-null, thereby any point inside this region is pushed/forced toward 0 by a gradient-descent technique. In Figure 2.2, for example, there are four plateaus in which $F_\beta(\mathbf{w}) = 2$, for any $\mathbf{w}$ belonging to the interior of these plateaus and, consequently, $\nabla F_\beta(\mathbf{w}) = 0$ for these $\mathbf{w}$'s. The remaining points of the parameter space, excluding the global minimum and the local minima, are such that $\nabla F_\beta(\mathbf{w}) \neq 0$ and, therefore, they belong to the zero-attraction region. In ultimate analysis, the choice of β concerns the size of the zero-attraction region, that is, it defines a threshold for the magnitude of the coefficients; coefficients below this threshold are pushed to 0, whereas the ones that are above this threshold remain unaltered.

Since $F_\beta(\mathbf{w})$ treats all entries w_i uniformly, it is easier to set β based on the univariate case depicted in Figure 2.1. Observe in Figures 2.1(c) and 2.1(d) that, for large values of β, the gradient $\nabla F_\beta(x)$ converges to 0 very fast as $|x|$ grows, meaning that the zero-attraction interval is short. On the other hand, the zero-attraction interval expands as β decreases. Thus, to set β properly, one must have some prior knowledge about the magnitude of the optimal coefficients. More precisely, one must have a clue about the magnitudes of the

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

relevant and irrelevant coefficients. Ideally, β should be chosen such that the irrelevant coefficients fall inside the zero-attraction interval, whereas the relevant coefficients fall outside such range, thus not being affected by F_β.

When dealing with sparse vectors, it is easy to set β because we know that the irrelevant coefficients are precisely equal to 0 and, therefore, a small zero-attraction interval suffices. Although high values of β could be used in this case, it is preferable to use moderate values of β, like $\beta \in [5, 10]$ (typically $\beta = 5$ [11]), because smoothness is important to guarantee the effectiveness of gradient-based optimization methods, as previously explained.

For compressible signals, on the other hand, the task of choosing β is more complicated. In this case, one should set β such that the low magnitude coefficients are pushed to 0, but the relevant (high magnitude) coefficients should not be severely affected. As a simple example, suppose that the optimal coefficients have some negligible entries with magnitudes ranging from 0 to 0.5, whereas its relevant entries are greater than 1 in magnitude. In this example, considering the LF function depicted in Figure 2.1(a), one should not use $\beta = 50$. Actually, it would be better to choose β equal to 2 or 5 to guarantee that coefficients with magnitudes up to 0.5 are pushed to 0 while producing little or no effect in the relevant coefficients.

2.2.5 Remarks

Let us summarize the main points about sparsity-promoting functions covered in this section. First, we learned that although the ℓ_0 norm is the natural function to model sparsity, it has critical issues, due to its discontinuity, that prevent its use in practical applications. Then, we studied three functions that are frequently used to replace the ℓ_0 norm in optimization problems, namely, the ℓ_1 norm, the reweighted ℓ_1 norm $F_{r\ell_1}$, and the ℓ_0-norm approximation F_β, which are illustrated in Figure 2.3 for the case of a scalar independent variable $x \in \mathbb{R}$. These three functions are continuous and almost everywhere differentiable, thus allowing the use of gradient-based optimization techniques, meaning that the training technique is simple, an important feature when dealing with high-dimensional problems. However, only the ℓ_1 norm is convex, being, therefore, easier to deal with than the other two. On the other hand, the ℓ_1 norm has three significant drawbacks: (i) heavy penalization of the large coefficients, (ii) its gradient pushes all coefficients to 0 with equal strength which results in the shrinkage of relevant coefficients, and (iii) high sparsity degree is necessary to obtain its benefits. Next, we explained that $F_{r\ell_1}$ mitigates these issues, while F_β eliminates them. On the other hand, both of these functions are nonconvex, but only F_β satisfies the saturation property, as can be verified in Figure 2.3, which is key to solving the drawbacks (i) and (ii) mentioned above. These functions have been compared experimentally in some works, and the results are usually the same: The ℓ_0-norm approximation F_β provides the best performance, followed by $F_{r\ell_1}$, and the ℓ_1 norm comes in third place [11, 21, 22, 24, 25].

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

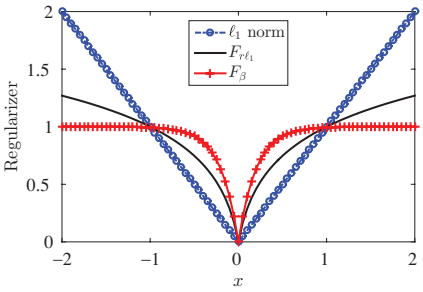


Figure 2.3 Sparsity-promoting regularizers applied on $x \in \mathbb{R}$: ℓ_1 norm; $F_{r\ell_1}$ with $\epsilon = 10$ [20]; F_β as the Laplace Function with $\beta = 5$ [11].

However, when working with nonconvex regularizers (that is, regularization functions), the possibility of introducing some undesired local minima or, even worse, changing the position of the global minimum exists. In this subsection, we elaborate more on this.

In Subsection 2.2.4, we showed an example of F_β surface containing some local minima (see Figure 2.2) and explained that by reducing β we could make these local minima as further from the origin as we want. We also mentioned the SL0 algorithm [29] as an efficient way to minimize $F_\beta(\mathbf{w})$, subject to some constraints on the parameters $\mathbf{w}$, using the steepest-descent method.

In adaptive filtering, however, F_β is usually employed as a regularizer, that is, a penalty function applied to the original objective function that we want to minimize, leading to the following regularized objective function $\mathcal{R}$:

$$\mathcal{R} = \begin{pmatrix} \text{Original objective} \\ \text{function} \end{pmatrix} + \alpha \begin{pmatrix} \text{Sparsity-promoting} \\ \text{regularizer} \end{pmatrix}, \tag{2.15}$$

where $\alpha \in \mathbb{R}_+$ is the *regularization parameter*, responsible for determining the weight given to the regularizer. While the original objective function aims to find the coefficients $\mathbf{w}$ that best explain/model the data, the sparsity-promoting regularizer aims to sparsify $\mathbf{w}$, by maximizing the number of entries of $\mathbf{w}$ equal to 0. Since the regularization is used to help in the minimization of the original objective function (and not the other way around), the regularization must be much weaker in terms of magnitude; thus, α must be small. If this rule is not satisfied, then the regularization will have a high impact on $\mathcal{R}$ and will possibly generate a local or global minimum at $\mathbf{w} = \mathbf{0}$. It is worth noticing that, once again, the saturation property of F_β helps us since we know its maximum value, that is, for $\mathbf{w} \in \mathbb{R}^{N+1}$, $\max F_\beta(\mathbf{w}) = N + 1$. This implies that, for any point $\mathbf{w}$ in the parameter space, the value of $\mathcal{R}$ is equal to the value of the original objective function plus $\alpha \times (N + 1)$, in the worst case (i.e., this corresponds to the most significant change on the original objective function due to F_β). Therefore, a sufficient condition to guarantee that no undesired global minimum appears at $\mathbf{w} = \mathbf{0}$ is

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

$$\begin{pmatrix}\text{Original objective function}\\ \text{evaluated at } \mathbf{w}=\mathbf{0}\end{pmatrix} > \begin{pmatrix}\text{Original objective function}\\ \text{evaluated at } \mathbf{w}^\star\end{pmatrix} + \alpha(N+1),$$

where $\mathbf{w}^\star$ is the minimizer of the original objective function. Observe that $F_\beta(\mathbf{0}) = 0$, which means that $\mathcal{R}$ and the original objective function are equal at $\mathbf{w} = \mathbf{0}$, whereas $\alpha(N+1)$ corresponds to the maximum value of $\alpha F_\beta(\mathbf{w}^\star)$, which occurs only when $\mathbf{w}^\star$ does not have coefficients close to 0 (i.e., within the zero-attraction region).

In addition, adaptive filtering algorithms are very often derived based on an original objective function that is convex, for example, the MSE or the minimum disturbance criterion, both introduced in Chapter 1. The next example illustrates that by choosing α correctly, the regularized objective function $\mathcal{R}$ may remain convex, albeit F_β is nonconvex.

Example 2.1 (The Role of α)

Let us consider an example in which the original objective function is the MSE. It is widely known that the MSE surface is a paraboloid in the parameter space [30], and we assume that it has circular contours as depicted in Figure 2.4(a).[6] In addition, let us assume that the sparsity-promoting regularizer is the ℓ_0-norm approximation given in Equation (2.12a), that is, the LF with $\beta = 5$, and the minimum MSE solution is $\mathbf{w}_\text{o} = [1\ 0]^\text{T}$, marked with a circle in Figure 2.4, which depicts the surface of $\mathcal{R}$ for different values of the regularization parameter α.

In Figure 2.4(a), we have $\alpha = 0$ and, therefore, we observe only the MSE contours. As α increases from 0 to 0.1, we observe that the pattern of the contours change gradually, becoming similar to ellipses with minor axis aligned with the vertical coordinate w_1. This pattern is the effect of the regularizer F_β over the coordinate w_1, corresponding to the zero-entry of $\mathbf{w}_\text{o}$, while the coordinate w_0, which corresponds to the relevant/nonzero entry of $\mathbf{w}_\text{o}$, is almost not affected. In Figure 2.4(c), we set $\alpha = 1$ so that the regularization starts to compete against the original objective function, thus generating a local minimum at $\mathbf{w} = \mathbf{0}$, represented with a triangle in the figure. Finally, in Figure 2.4(d), we use $\alpha = 10$ to illustrate a case in which the regularization becomes dominant in $\mathcal{R}$, leading to a global minimum at $\mathbf{w} = \mathbf{0}$.

Observe that the colorbar ranges in Figure 2.4 can be used to indicate a bad choice of α. Indeed, when α is properly chosen, like in Figure 2.4(b), this range is very similar to the range obtained when $\alpha = 0$, see Figure 2.4(a), meaning that the regularization is not competing against the original objective function. On the other hand, if α is such that the regularization is capable of changing

[6] The contours of a surface correspond to the geometric places in which the surface has constant values. The MSE surface exhibits circular contours for uncorrelated inputs whose covariance matrix is $\sigma_x^2\mathbf{I}$, where σ_x^2 is the variance of the input signal and $\mathbf{I}$ is the identity matrix.

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

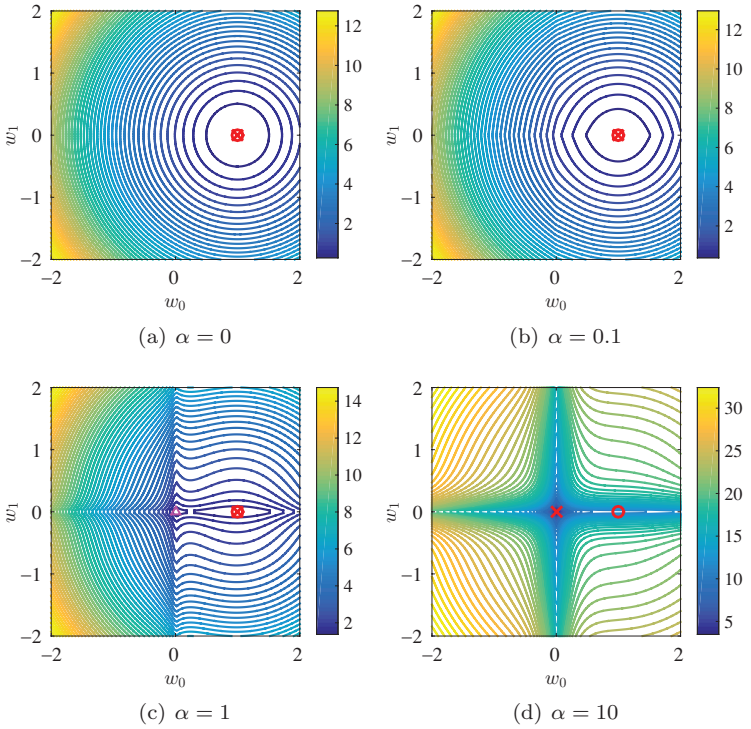


Figure 2.4 Contours of $\mathcal{R}$ for different values of the regularization parameter α: (a) $\alpha = 0$ (just the MSE contours); (b) $\alpha = 0.1$; (c) $\alpha = 1$; and (d) $\alpha = 10$. The circle denotes the minimum MSE solution $\mathbf{w}_o = [1\ 0]^T$, the cross is placed at the global minimum of $\mathcal{R}$, and the local minimum is represented by a triangle.

the colorbar range, like in Figures 2.4(c) and 2.4(d), then α might be too high and it would be wise reducing its value.

Note: The values of α used in Figure 2.4 are just for illustration purposes and should not be taken as guidelines. Usually, the choice of α changes from problem to problem. For instance, in the case of $\alpha = 0.1$, a local minimum might appear (just like in Figure 2.4(c)) if the value of β is increased or if the variance of the input signal σ_x^2 is decreased; in this case, one would have to reduce α to avoid the local minimum issue. Mathematically, to guarantee that $\mathcal{R}$ preserves the convexity of the original optimization problem, one must calculate its Hessian matrix and force it to be positive definite by selecting α and β properly.

2.3 Algorithms Employing Sparsity-Promoting Regularizations

In this section, we apply the sparsity-promoting regularization functions studied in Section 2.2 to some classical algorithms in order to enable them to exploit

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

Table 2.1 Summary of the sparsity-promoting regularizers $\mathcal{F}$ and their gradients

	$\mathcal{F}(\mathbf{w})$	i-th entry of $\mathbf{f}(\mathbf{w}) \triangleq \nabla\mathcal{F}(\mathbf{w})$
ℓ_1 norm	$\|\mathbf{w}\|_1$, see Equation (2.2)	$f_{\ell_1}(w_i) = \text{sign}(w_i)$
Reweighted ℓ_1 ($r\ell_1$) norm	$F_{r\ell_1}(\mathbf{w})$, see Equation (2.5)	$f_{r\ell_1}(w_i) = \dfrac{\text{sign}(w_i)}{1+\epsilon\lvert w_i\rvert}$
ℓ_0-norm approx.	$F_\beta(\mathbf{w})$, see Equation (2.12)	$f_\beta(w_i) = \begin{cases} \beta\text{sign}(w_i)\mathrm{e}^{-\beta\lvert w_i\rvert} & \text{for the LF} \\ \dfrac{\beta\text{sign}(w_i)}{(1+\beta\lvert w_i\rvert)^2} & \text{for the GMF} \end{cases}$

the sparsity of a vector/signal. To provide a concise and unified exposition of regularization-based algorithms, we use the symbol $\mathcal{F}$ to denote any sparsity-promoting regularizer, and we postpone its choice as much as possible aiming at emphasizing the similarities among the several algorithms. For the reader's convenience, Table 2.1 summarizes the possible choices of $\mathcal{F}$ as well as their gradient expressions, covered in detail in Section 2.2.

The classical algorithms considered in this section are the LMS, the NLMS, and the AP algorithms, each of which was briefly discussed in Chapter 1. To generate the regularized versions of these algorithms, we follow the standard procedure: We take the original objective function used to derive each of these algorithms and we add $\mathcal{F}(\mathbf{w})$ as a penalty function. We also define the gradient of $\mathcal{F}(\mathbf{w})$ as

$$\nabla\mathcal{F}(\mathbf{w}) \triangleq \mathbf{f}(\mathbf{w}) = [f(w_0)\ f(w_1)\ \ldots\ f(w_N)]^{\mathrm{T}} \in \mathbb{R}^{N+1}. \tag{2.16}$$

After selecting the function $\mathcal{F}$, we include a subscript on $\mathbf{f}$ to inform of this choice. For example, if we choose $\mathcal{F} = F_\beta(\mathbf{w})$, then its gradient vector will be denoted by $\mathbf{f}_\beta(\mathbf{w})$ whose i-th entry is $f_\beta(w_i)$, as given in Table 2.1.

In the adaptive filtering literature, the prefixes *zero-attractor* (ZA), *reweighted zero-attractor* (RZA), and ℓ_0 are very often used to indicate that the ℓ_1 norm, the $r\ell_1$ norm, and the ℓ_0-norm approximation, respectively, were chosen as the sparsity-promoting regularization. For the LMS algorithm, for example, this leads to the following three sparsity-aware LMS-based algorithms: the *zero-attractor* LMS (ZA-LMS), the *reweighted zero-attractor* LMS (RZA-LMS), and the ℓ_0-*norm approximation* LMS (ℓ_0-LMS) algorithms.

2.3.1 Regularized LMS Algorithms

As explained in Chapter 1, the LMS algorithm uses a gradient-descent method to minimize the squared error $|e(k)|^2$, which can be regarded as an instantaneous and stochastic approximation of the MSE [30]. Thus, the regularized

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

objective function corresponding to the LMS algorithm is given by

$$\mathcal{R}_{\text{LMS}}(\mathbf{w}(k)) \triangleq |e(k)|^2 + \alpha\mathcal{F}(\mathbf{w}(k)), \tag{2.17}$$

where $\mathcal{F}$ is any of the sparsity-promoting regularizers given in Table 2.1 and $\alpha \in \mathbb{R}_+$ is the regularization parameter, which is usually chosen as a small number in order for the regularization to be less important than the original objective function in the regularized problem $\mathcal{R}_{\text{LMS}}$, as explained in the Subsection 2.2.5.

Since the LMS algorithm employs a gradient-descent method, its recursion depends on the gradient of $\mathcal{R}_{\text{LMS}}$ with respect to $\mathbf{w}(k)$, denoted by $\nabla\mathcal{R}_{\text{LMS}}(\mathbf{w}(k))$, and is given by

$$\begin{aligned}\mathbf{w}(k+1) &= \mathbf{w}(k) - \mu\ \nabla\mathcal{R}_{\text{LMS}}(\mathbf{w}(k)) \\ &= \mathbf{w}(k) - \mu\ \nabla|e(k)|^2 - \mu\alpha\ \nabla\mathcal{F}(\mathbf{w}(k)) \\ &= \mathbf{w}(k) + \underbrace{2\mu e(k)\mathbf{x}(k)}_{\text{LMS correction}} - \underbrace{\mu\alpha\mathbf{f}(\mathbf{w}(k))}_{\text{Regularization correction}},\end{aligned} \tag{2.18}$$

where we recognize a correction term corresponding to the standard LMS algorithm, whereas the regularization introduces a new correction term that encourages the filter coefficients $\mathbf{w}(k+1)$ to be sparse. The expressions for the gradient vector $\mathbf{f}(\mathbf{w}(k)) \triangleq \nabla\mathcal{F}(\mathbf{w}(k))$ can be found in Table 2.1 for the ℓ_1 norm, the $r\ell_1$ norm, and the ℓ_0-norm approximation. The LMS-based algorithms employing the ℓ_1 norm, the $r\ell_1$ norm, and the ℓ_0-norm approximation are known as ZA-LMS [23], RZA-LMS [23], and ℓ_0-LMS [25], respectively.[7] These algorithms are summarized in Algorithm 2.2.

Algorithm 2.2 The sparsity-aware LMS algorithms

Initialization:

$\mathbf{x}(0) = \mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\text{T}}$

choose μ and α as small positive numbers

choose the regularizer $\mathcal{F}$ (and related parameters):

$$\text{If } \mathcal{F} = \begin{cases} \|\mathbf{w}(k)\|_1, & \text{then we have the ZA-LMS algorithm [23]} \\ F_{r\ell_1}(\mathbf{w}(k)), & \text{then we have the RZA-LMS algorithm [23]} \\ F_\beta(\mathbf{w}(k)) & \text{then we have the } \ell_0\text{-LMS algorithm [25]} \end{cases}$$

For $k \geq 0$ (i.e., for every iteration) **do**

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

Compute $\mathbf{f}(\mathbf{w}(k))$ corresponding to the selected $\mathcal{F}$ (see Table 2.1)

$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu e(k)\mathbf{x}(k) - \mu\alpha\mathbf{f}(\mathbf{w}(k))$

[7] The ℓ_0-LMS algorithm proposed in [25] uses the LF as F_β. However, instead of using the gradient $\mathbf{f}_\beta(\mathbf{w}(k))$, whose entries are specified in Table 2.1, it uses the Taylor series to approximate the exponential function by a first-order polynomial, for small values of $|w_i(k)|$. In this case, the algorithm exchanges quality of approximation (of the ℓ_0 norm), thus performance, for reduced computational burden.

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

Note: When we provide the step-by-step of an algorithm, like in Algorithm 2.2, we are focusing on didactic rather than computational efficiency. For example, we could avoid two multiplications at every iteration by defining and storing the auxiliary variables $\mu' = 2\mu$ and $\alpha' = \mu\alpha$ during the initialization stage.

2.3.2 Regularized AP Algorithms

Let us start by revisiting the notation of the main variables related to the AP algorithm (for more details, check Chapter 1):

$$\begin{array}{rcll} \mathbf{x}(k) & = & [x(k)\ x(k-1)\ \ldots\ x(k-N)]^{\mathrm{T}} & \in \mathbb{R}^{N+1}, \\ \mathbf{X}(k) & = & [\mathbf{x}(k)\ \mathbf{x}(k-1)\ \ldots\ \mathbf{x}(k-L)] & \in \mathbb{R}^{(N+1)\times(L+1)}, \\ \mathbf{w}(k) & = & [w_0(k)\ w_1(k)\ \ldots\ w_N(k)]^{\mathrm{T}} & \in \mathbb{R}^{N+1}, \\ \mathbf{d}(k) & = & [d(k)\ d(k-1)\ \ldots\ d(k-L)]^{\mathrm{T}} & \in \mathbb{R}^{L+1}, \\ \mathbf{e}(k) & = & [e_0(k)\ e_1(k)\ \ldots\ e_L(k)]^{\mathrm{T}} & \in \mathbb{R}^{L+1}, \end{array} \tag{2.19}$$

where $\mathbf{x}(k)$ is the input vector (in tapped-delay line format), $\mathbf{X}(k)$ is the input matrix that includes the current input vector $\mathbf{x}(k)$ and also L previous input vectors, $\mathbf{w}(k)$ is the adaptive filter coefficient vector, $\mathbf{d}(k)$ is the desired vector, and $\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{X}^{\mathrm{T}}(k)\mathbf{w}(k)$ is the error vector. In addition, N is the order of the adaptive filter, and L is the data reuse factor, that is, the amount of data from previous iterations that is to be used in the current iteration. Observe that $e_0(k)$, the 0-th entry of $\mathbf{e}(k)$, is computed using the data from the current iteration $(d(k), \mathbf{x}(k))$ and, therefore, it is equivalent to $e(k)$ in the LMS algorithm.

As explained in Chapter 1, the AP algorithm is derived by minimizing the minimum disturbance criterion (or principle) subject to some constraints related to the *a posteriori* errors being equal to 0. Thus, the regularized AP algorithms solve the following optimization problem

$$\begin{aligned} &\text{minimize } \mathcal{R}_{\mathrm{AP}}(\mathbf{w}(k+1)) \triangleq \|\mathbf{w}(k+1) - \mathbf{w}(k)\|_2^2 + \alpha\mathcal{F}(\mathbf{w}(k+1)), \\ &\text{subject to } \mathbf{d}(k) - \mathbf{X}^{\mathrm{T}}(k)\mathbf{w}(k+1) = \mathbf{0}, \end{aligned} \tag{2.20}$$

where $\mathcal{R}_{\mathrm{AP}}(\mathbf{w}(k+1))$ represents the regularized objective function related to the AP algorithm. As explained in Subsection 2.3.1, $\mathcal{F}$ is any of the sparsity-promoting regularizers given in Table 2.1 and $\alpha \in \mathbb{R}_+$ is the regularization parameter.

In order to solve this optimization problem, we form the Lagrangian $\mathcal{L}$ as

$$\begin{aligned} \mathcal{L}(\mathbf{w}(k+1), \boldsymbol{\lambda}) =& \|\mathbf{w}(k+1) - \mathbf{w}(k)\|_2^2 + \alpha\mathcal{F}\left(\mathbf{w}(k+1)\right) \\ &+ \boldsymbol{\lambda}^{\mathrm{T}}\left[\mathbf{d}(k) - \mathbf{X}^{\mathrm{T}}(k)\mathbf{w}(k+1)\right], \end{aligned} \tag{2.21}$$

which is a function of both the parameters $\mathbf{w}(k+1)$ and the Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^{L+1}$. Then, we differentiate $\mathcal{L}$ with respect to $\mathbf{w}(k+1)$ and $\boldsymbol{\lambda}$ and equate

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

the results to 0 (i.e., $\nabla\mathcal{L} = \mathbf{0}$) to obtain

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mathbf{X}(k)\frac{\boldsymbol{\lambda}}{2} - \frac{\alpha}{2}\,\nabla\mathcal{F}\left(\mathbf{w}(k+1)\right), \tag{2.22}$$

$$\mathbf{X}^{\mathrm{T}}(k)\mathbf{w}(k+1) = \mathbf{d}(k), \tag{2.23}$$

respectively. Then, the left multiplication of Equation (2.22) by $\mathbf{X}^{\mathrm{T}}(k)$ followed by the substitution of Equation (2.23) into the resulting equation leads to

$$\begin{aligned}\frac{\boldsymbol{\lambda}}{2} &= \left(\mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k)\right)^{-1}\mathbf{e}(k)\\ &\quad+\frac{\alpha}{2}\left(\mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k)\right)^{-1}\mathbf{X}^{\mathrm{T}}(k)\,\nabla\mathcal{F}\left(\mathbf{w}(k+1)\right),\end{aligned} \tag{2.24}$$

where it is assumed that $\mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k)$ is invertible. By substituting Equation (2.24) into Equation (2.22) we obtain

$$\begin{aligned}\mathbf{w}(k+1) = \mathbf{w}(k) &+ \mathbf{X}(k)\left(\mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k)\right)^{-1}\mathbf{e}(k)\\ &+\frac{\alpha}{2}\left[\mathbf{X}(k)\left(\mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k)\right)^{-1}\mathbf{X}^{\mathrm{T}}(k)-\mathbf{I}\right]\mathbf{f}(\mathbf{w}(k+1)),\end{aligned} \tag{2.25}$$

where $\mathbf{f}(\mathbf{w}(k+1)) \triangleq \nabla\mathcal{F}\left(\mathbf{w}(k+1)\right)$, as already defined. To transform the previous equation in a recursion, we replace $\mathbf{f}(\mathbf{w}(k+1))$ with $\mathbf{f}(\mathbf{w}(k))$ to obtain the recursion of the regularized AP algorithms:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \underbrace{\mu\mathbf{X}(k)\mathbf{S}(k)\mathbf{e}(k)}_{\text{AP correction}} + \underbrace{\mu\frac{\alpha}{2}\left[\mathbf{X}(k)\mathbf{S}(k)\mathbf{X}^{\mathrm{T}}(k)-\mathbf{I}\right]\mathbf{f}(\mathbf{w}(k))}_{\text{Regularization correction}}, \tag{2.26}$$

where, just like in the classical AP algorithm, we included a relaxation parameter μ that acts like a step size, and we defined $\mathbf{S}(k) \triangleq \left(\mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k) + \gamma\mathbf{I}\right)^{-1}$, where $\gamma \in \mathbb{R}_+$ is a small positive number used to prevent matrix $\mathbf{S}(k)$ from becoming singular or ill-conditioned. In Equation (2.26), we recognize a correction term corresponding to the standard AP algorithm, whereas the regularization introduces a new correction term that encourages the filter coefficients $\mathbf{w}(k+1)$ to be sparse.

Once again, the expressions for the gradient vector $\mathbf{f}(\mathbf{w}(k))$ can be found in Table 2.1 for the ℓ_1 norm, the $r\ell_1$ norm, and the ℓ_0-norm approximation. The AP-based algorithms employing the ℓ_1 norm, the $r\ell_1$ norm, and the ℓ_0-norm approximation are known as *zero-attractor* AP (ZA-AP) [31], *reweighted zero-attractor* AP (RZA-AP) [31], and ℓ_0*-norm approximation* AP (ℓ_0-AP) [21], respectively.[8] These algorithms are summarized in Algorithm 2.3.

[8] Actually, the authors in [21, 31] use ZA-APA and RZA-APA, instead of ZA-AP and RZA-AP, where APA stands for *affine projection algorithm*. Moreover, the herein called ℓ_0-AP algorithm was proposed in [21] by the name of *affine projection algorithm for sparse system identification*, abbreviated both as AP-SSI or APA-SSI in [11, 28]. In this chapter, we opted for using ℓ_0-AP instead of APA-SSI since the latter name is not informative of which sparsity-promoting regularizer is being used.

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

Algorithm 2.3 The sparsity-aware AP algorithms

Initialization:

$\mathbf{x}(0) = \mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\mathrm{T}}$

choose $\mu \in (0, 1]$

choose α and γ as small positive numbers

choose the regularizer $\mathcal{F}$ (and related parameters):

$$\text{If } \mathcal{F} = \begin{cases} \|\mathbf{w}(k)\|_1, & \text{then we have the ZA-AP algorithm [31]} \\ F_{r\ell_1}(\mathbf{w}(k)), & \text{then we have the RZA-AP algorithm [31]} \\ F_{\beta}(\mathbf{w}(k)) & \text{then we have the } \ell_0\text{-AP algorithm [21]} \end{cases}$$

For $k \geq 0$ (i.e., for every iteration) **do**

$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{X}^{\mathrm{T}}(k)\mathbf{w}(k)$

$\mathbf{S}(k) = \left(\mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k) + \gamma\mathbf{I}\right)^{-1}$

Compute $\mathbf{f}(\mathbf{w}(k))$ corresponding to the selected $\mathcal{F}$ (see Table 2.1)

$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu\mathbf{X}(k)\mathbf{S}(k)\mathbf{e}(k) + \mu\frac{\alpha}{2}\left[\mathbf{X}(k)\mathbf{S}(k)\mathbf{X}^{\mathrm{T}}(k) - \mathbf{I}\right]\mathbf{f}(\mathbf{w}(k))$

It is interesting to interpret the optimization problem given in Equation (2.20) from the geometric point of view. Its solution $\mathbf{w}(k+1)$ must satisfy the constraint $\mathbf{X}^{\mathrm{T}}(k)\mathbf{w}(k+1) = \mathbf{d}(k)$, meaning that $\mathbf{w}(k+1)$ must lie on the intersection of $(L+1)$ hyperplanes, denoted by $\Pi_{(L+1)}$. If $\alpha = 0$, then $\mathbf{w}(k+1)$ is generated as an orthogonal projection of $\mathbf{w}(k)$ onto $\Pi_{(L+1)}$, which is the minimum Euclidean norm solution; this corresponds to the standard AP update (with $\mu = 1$). For $\alpha \neq 0$, the update process can be explained as a two-step procedure: (i) the standard AP update transports $\mathbf{w}(k)$ to a point $\mathbf{w}' \in \Pi_{(L+1)}$ and (ii) the regularization correction maps $\mathbf{w}'$ to $\mathbf{w}(k+1) \in \Pi_{(L+1)}$ by encouraging $\mathbf{w}(k+1)$ to be sparser than $\mathbf{w}'$, but still restricted to belong to $\Pi_{(L+1)}$. The fact that the sparse solution $\mathbf{w}(k+1)$ is confined to $\Pi_{(L+1)}$ can be seen as an advantage since it facilitates the choice of α (we postpone this explanation to Subsection 2.3.3), but it can also be seen as a disadvantage since it reduces the length of $\mathbf{f}(\mathbf{w}(k))$, as the length of a vector is always greater than or equal to the length of its projection, and requires more computations, as explained in [11, 21]. This issue motivated the development of a new algorithm, which eliminates the matrix $\mathbf{X}(k)\mathbf{S}(k)\mathbf{X}^{\mathrm{T}}(k)$, leading to the following recursion:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \underbrace{\mu\mathbf{X}(k)\mathbf{S}(k)\mathbf{e}(k)}_{\text{AP correction}} - \underbrace{\mu\frac{\alpha}{2}\mathbf{f}(\mathbf{w}(k))}_{\text{Regularization correction}} . \tag{2.27}$$

Observe that, for $\mu = 1$ and $\gamma = 0$, that is, eliminating the constants that were artificially introduced to increase the algorithm robustness, the $\mathbf{w}(k+1)$ in Equation (2.26) satisfies the constraint in Equation (2.20), but the $\mathbf{w}(k+1)$ in Equation (2.27) does not. Therefore, algorithms employing the recursion given in Equation (2.27) should not be called AP. Here, we call them *quasi-AP* (qAP) following the work that first noticed this fact [21]. The qAP algorithms employing sparsity-promoting regularizations are summarized in Algorithm 2.4.

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

Algorithm 2.4 The sparsity-aware qAP algorithms [11, 21]

Initialization:

$\mathbf{x}(0) = \mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\mathrm{T}}$

choose $\mu \in (0, 1]$

choose α and γ as small positive numbers

choose the regularizer $\mathcal{F}$ (and related parameters):

$$\text{If } \mathcal{F} = \begin{cases} \|\mathbf{w}(k)\|_1, & \text{then we have the ZA-qAP algorithm} \\ F_{r\ell_1}(\mathbf{w}(k)), & \text{then we have the RZA-qAP algorithm} \\ F_{\beta}(\mathbf{w}(k)) & \text{then we have the } \ell_0\text{-qAP algorithm [21]} \end{cases}$$

For $k \geq 0$ (i.e., for every iteration) **do**

$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{X}^{\mathrm{T}}(k)\mathbf{w}(k)$

$\mathbf{S}(k) = \left(\mathbf{X}^{\mathrm{T}}(k)\mathbf{X}(k) + \gamma\mathbf{I}\right)^{-1}$

Compute $\mathbf{f}(\mathbf{w}(k))$ corresponding to the selected $\mathcal{F}$ (see Table 2.1)

$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu\mathbf{X}(k)\mathbf{S}(k)\mathbf{e}(k) - \mu\frac{\alpha}{2}\mathbf{f}(\mathbf{w}(k))$

Hint: If α is properly chosen, both AP and qAP algorithms provide similar performance in terms of convergence speed and MSE, but the qAP algorithm is more efficient. However, when a qAP algorithm is used with nonstationary input $\mathbf{x}(k)$, the problem of choosing α is considerably more difficult and maybe even impossible. This issue will be explained in Subsection 2.3.3.

Note: To the best of our knowledge, there is no article proposing the ZA-qAP and RZA-qAP algorithms. We opted for introducing them here only to maintain the pattern we have been following.

2.3.3 Regularized NLMS Algorithms

As explained in Chapter 1, the NLMS algorithm is a particular case of the AP algorithm, where $L = 0$, that is, only the data from the current iteration is used. Therefore, in this subsection, we present only its related optimization problem and the algorithms recursions. The derivation of these recursions is omitted here because it follows precisely the same steps used in Subsection 2.3.2.

The regularized NLMS algorithms solve the following optimization problem:

$$\begin{aligned} &\text{minimize } \mathcal{R}_{\mathrm{NLMS}}(\mathbf{w}(k+1)) \triangleq \|\mathbf{w}(k+1) - \mathbf{w}(k)\|_2^2 + \alpha\mathcal{F}(\mathbf{w}(k+1)), \\ &\text{subject to } d(k) - \mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k+1) = 0, \end{aligned} \tag{2.28}$$

where $\mathcal{R}_{\mathrm{NLMS}}(\mathbf{w}(k+1))$ represents the regularized objective function related to the NLMS algorithm, $\mathcal{F}$ is any of the sparsity-promoting regularizers given in Table 2.1, and $\alpha \in \mathbb{R}_+$ is the regularization parameter.

The solution to the optimization problem in Equation (2.28) leads to the following recursion:

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

Algorithm 2.5 The sparsity-aware NLMS algorithms

Initialization:

$\mathbf{x}(0) = \mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\mathrm{T}}$

choose $\mu \in (0, 1]$

choose α and γ as small positive numbers

choose the regularizer $\mathcal{F}$ (and related parameters):

$$\text{If } \mathcal{F} = \begin{cases} \|\mathbf{w}(k)\|_1, & \text{then we have the ZA-NLMS algorithm [31]} \\ F_{r\ell_1}(\mathbf{w}(k)), & \text{then we have the RZA-NLMS algorithm [31]} \\ F_{\beta}(\mathbf{w}(k)) & \text{then we have the } \ell_0\text{-NLMS algorithm [21]} \end{cases}$$

For $k \geq 0$ (i.e., for every iteration) **do**

$e(k) = \mathbf{d}(k) - \mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k)$

Compute $\mathbf{f}(\mathbf{w}(k))$ corresponding to the selected $\mathcal{F}$ (see Table 2.1)

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mu e(k)}{\|\mathbf{x}(k)\|_2^2 + \gamma}\mathbf{x}(k) + \mu\frac{\alpha}{2}\left[\frac{\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)}{\|\mathbf{x}(k)\|_2^2 + \gamma} - \mathbf{I}\right]\mathbf{f}(\mathbf{w}(k))$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \underbrace{\frac{\mu e(k)}{\|\mathbf{x}(k)\|_2^2 + \gamma}\mathbf{x}(k)}_{\text{NLMS correction}} + \underbrace{\mu\frac{\alpha}{2}\left[\frac{\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)}{\|\mathbf{x}(k)\|_2^2 + \gamma} - \mathbf{I}\right]\mathbf{f}(\mathbf{w}(k))}_{\text{Regularization correction}}, \quad (2.29)$$

where, as explained in Subsection 2.3.2, μ can be understood both as a relaxation factor, since it actually relaxes the constraint in Equation (2.28), or as a step size and should be chosen as $\mu \in (0, 1]$, and $\gamma \in \mathbb{R}_+$ is a small positive number used to avoid numerical problems that occur when $\|\mathbf{x}(k)\|_2^2 \to 0$. Algorithm 2.5 summarizes the sparsity-aware NLMS algorithms employing different regularizations.

Following the same reasoning that motivated the qAP algorithms in Subsection 2.3.2, for the NLMS algorithm we can eliminate the matrix $\frac{\mathbf{x}(k)\mathbf{x}^{\mathrm{T}}(k)}{\|\mathbf{x}(k)\|_2^2+\gamma}$ in order to reduce the number of arithmetic operations required by the recursion and also to let the sparsity-promoting term $\mathbf{f}(\mathbf{w}(k))$ to be unconstrained, leading to the recursion of the so-called *quasi-NLMS* (qNLMS) algorithms given by

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \underbrace{\frac{\mu e(k)}{\|\mathbf{x}(k)\|_2^2 + \gamma}\mathbf{x}(k)}_{\text{NLMS correction}} - \underbrace{\mu\frac{\alpha}{2}\mathbf{f}(\mathbf{w}(k))}_{\text{Regularization correction}}. \quad (2.30)$$

The qNLMS algorithms employing sparsity-promoting regularizations are summarized in Algorithm 2.6.

Note: The ℓ_0-qNLMS algorithm was proposed in [25] under the name of ℓ_0-NLMS algorithm, but this is not an accurate name since its update equation for $\mu = 1$ and $\gamma = 0$, see Equation (2.30), does not satisfy the constraint in Equation (2.28). Moreover, from the numerical point of view, just the NLMS correction term is normalized by the energy of $\mathbf{x}(k)$ (assuming γ is negligible compared to $\|\mathbf{x}(k)\|_2^2$), whereas the regularization correction is not normalized

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

Algorithm 2.6 The sparsity-aware qNLMS algorithms [11, 21]

Initialization:

$\mathbf{x}(0) = \mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\mathrm{T}}$

choose $\mu \in (0, 1]$

choose α and γ as small positive numbers

choose the regularizer $\mathcal{F}$ (and related parameters):

$$\text{If } \mathcal{F} = \begin{cases} \|\mathbf{w}(k)\|_1, & \text{then we have the ZA-qNLMS algorithm} \\ F_{r\ell_1}(\mathbf{w}(k)), & \text{then we have the RZA-qNLMS algorithm} \\ F_{\beta}(\mathbf{w}(k)) & \text{then we have the } \ell_0\text{-qNLMS algorithm [21, 25]} \end{cases}$$

For $k \geq 0$ (i.e., for every iteration) **do**

$e(k) = \mathbf{d}(k) - \mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k)$

Compute $\mathbf{f}(\mathbf{w}(k))$ corresponding to the selected $\mathcal{F}$ (see Table 2.1)

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mu e(k)}{\|\mathbf{x}(k)\|_2^2 + \gamma}\mathbf{x}(k) - \mu\frac{\alpha}{2}\mathbf{f}(\mathbf{w}(k))$$

and, as a result, it might be difficult to set α, especially in situations where the energy of $\mathbf{x}(k)$ may vary. Think of this as follows. There are two correction terms in the recursion: (i) the NLMS correction, which aims to model the data, thus reducing the squared error and (ii) the regularization correction, which encourages sparse solutions. For the NLMS versions described in Algorithm 2.5, since both correction terms are normalized by the energy of $\mathbf{x}(k)$, one needs to set just a single parameter α to determine the balance between modeling the data and finding sparse solutions. As already explained, modeling the data should be "more important" than sparsifying the solution, which justifies the use of low values of α. The point is, for the qNLMS versions given in Algorithm 2.6, as only one of these correction terms is normalized by $\|\mathbf{x}(k)\|_2^2$, the "degree of importance" related to this correction term is time-varying, whereas the "degree of importance" of the other term is constant and equal to α. For example, if at a given iteration k, the energy $\|\mathbf{x}(k)\|_2^2$ becomes very large, then the NLMS correction may become very small (tending to 0), whereas the regularization correction is unaltered. This situation would configure a case in which we put more emphasis on finding sparse solutions than on modeling the data and, as a consequence, it might generate local or global minimum points as discussed in Subsection 2.2.5.[9]

2.3.4 Numerical Experiments

Here, we compare the different sparsity-promoting regularizers considering that optimal coefficients have varying degrees of sparsity. The experiments consist

[9] Observe that this normalization issue in the qNLMS algorithm also happens in the qAP algorithm. We opted for explaining it here because the NLMS algorithm is easier to understand since it is closely related to the LMS algorithm and also because of the issue related to the name of the algorithm.

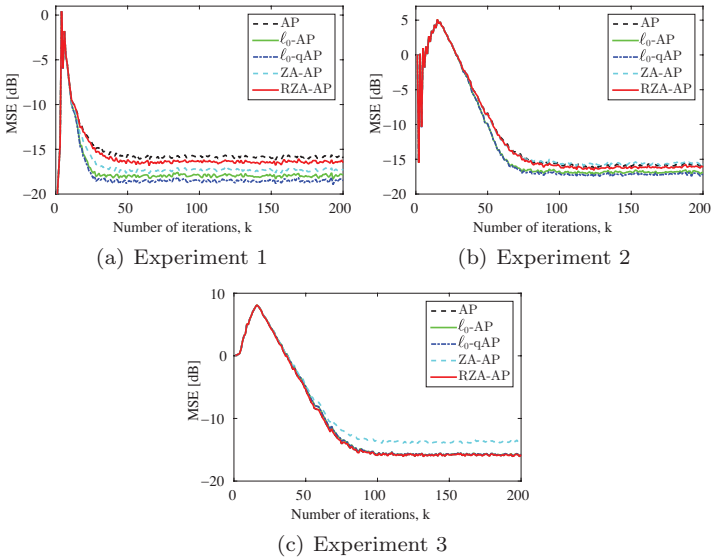
https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

of identifying an unknown system $\mathbf{w}_o$ comprised of 16 coefficients given by the following [21, 31]:

- In Experiment 1, we set its fourth tap equal to 1, whereas the others are equal to 0, thus the optimal coefficients have a very high sparsity degree.
- In Experiment 2, the odd taps are set to 1, while the even taps are equal to 0, thus $\mathbf{w}_o$ has a sparsity degree of 50%.
- In Experiment 3, all the 16 taps are equal to 1, that is, the sparsity degree in $\mathbf{w}_o$ is 0% (this kind of signal is also called *dispersive*).

As for the adaptive filter, the number of coefficients is also 16 and the following algorithms are tested: the AP, the ℓ_0-AP, the ℓ_0-qAP, the ZA-AP, and the RZA-AP. The parameters of these algorithms were set so that the algorithms differ only by the regularization function. Thus, we set the step size $\mu = 0.9$, the regularization factor $\gamma = 10^{-12}$, the data-reuse factor $L = 4$, F_β as the GMF with $\beta = 5$, and we use $\alpha = 5 \times 10^{-3}$ and $\epsilon = 100$, in accordance with the values used in [31]. Moreover, the reference signal $d(k)$ is assumed to be corrupted by an additive white Gaussian measurement noise with variance $\sigma_n^2 = 0.01$.

Figure 2.5 depicts the MSE results for Experiments 1–3. One can observe that all the algorithms exhibited similar convergence speeds, as they were adjusted with the same values of μ, γ, L, and α. In Experiment 1, depicted in Figure 2.5(a), all the sparsity-aware algorithms outperformed the AP algorithm due to the high sparsity degree involved in this experiment. Moreover, the ones



(a) Experiment 1 (b) Experiment 2

(c) Experiment 3

Figure 2.5 MSE learning curves for some sparsity-aware AP (and qAP) algorithms considering the optimal coefficients have different sparsity degrees: (a) Experiment 1 (very sparse); (b) Experiment 2 (moderately sparse); and (c) Experiment 3 (dispersive).

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

that use the ℓ_0-norm approximation (i.e., the ℓ_0-AP and the ℓ_0-qAP algorithms) achieved the best results. In Experiment 2, depicted in Figure 2.5(b), one can notice that the ℓ_0-AP and the ℓ_0-qAP algorithms were still better than the others, but the improvement over the AP algorithm was reduced since the unknown system is not very sparse. One should observe that, in this case, the algorithms employing the ℓ_1 norm and the $r\ell_1$ did not perform better than the AP algorithm. This indicates that these two regularizers require higher values of sparsity degree in order to bring some advantage. Finally, in Experiment 3, we tested these regularizers in a case where the unknown system does not have any coefficient equal to 0. One can observe that the algorithms employing the $r\ell_1$ norm and the ℓ_0-norm approximation achieved the same MSE results as the AP algorithm, that is, these regularizers were robust to the absence of sparsity (in the sense that they do not introduce any harm if the vector to be identified happens to be nonsparse/dispersive). This behavior is justified by the fact that the entries of the gradient vector corresponding to these two regularizers converge to 0 as $|w_i|$ grows, as can be verified in Table 2.1. The ZA-AP algorithm, on the other hand, employs the ℓ_1 norm, which shrinks the large coefficients leading to the worst MSE result, as depicted in Figure 2.5(c).

2.3.5 Further Readings

In this section, we applied the sparsity-promoting regularizers studied in Section 2.2 to the following three classical algorithms: the LMS, the NLMS, and the AP algorithms. We also provided some numerical experiments to illustrate the effect of each regularizer as the sparsity degree of the optimal coefficients vary. Clearly, our presentation focused on the regularizers, thus explaining how they can be combined to allow the exploitation of sparsity by the algorithms and also illustrating the benefits introduced by each of the regularizers. However, the reader must be aware that sparsity-promoting regularizations can be applied to many other algorithms, including those not belonging to the adaptive filtering field. Here, we provide some notes to complement the content of this section.

Still in the adaptive filtering context, sparsity-promoting regularizers have been combined with the recursive least squares (RLS) algorithm [32] and also with the set-membership (SM) framework. In [11], the SM versions of the ℓ_0-AP and ℓ_0-qAP algorithms were proposed (under the name of SSM-AP and QSSM-AP, respectively). These algorithms combine the advantages of the ℓ_0-norm approximation with the data selection scheme used in the SM approach, which confers robustness against uncertainties and noise [33–35]. Indeed, in the comprehensive set of simulations provided in [11], which encompasses many algorithms as well as sparse and compressible signals, these algorithms achieved remarkable MSE and misalignment results. Also, notice that in the same way we used the fact that the NLMS algorithm can be obtained from the AP algorithm

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

by setting the data-reuse factor as $L = 0$, sparsity-promoting versions of the binormalized LMS (BNLMS) algorithm can be derived by making $L = 1$.

Recently, the sparsity-promoting regularizers described in Section 2.2 have been applied to the so-called *feature adaptive filtering* [1, 2] to exploit the hidden sparsity in the optimal parameters. Although most works in this area employ the ℓ_1 norm due to its simplicity, the ℓ_0-norm approximation has proven to be more advantageous [36].

In the context of nonlinear adaptive filtering, sparsity-promoting regularizers have been used in Volterra filters [37]. In fact, the Volterra series provides an excellent basis to apply these regularizers, as the number of coefficients increase very fast with the filter and memory orders of the Volterra series [38, 39]. Indeed, since in practical applications the nonlinear components are often much fewer than the number of coefficients in a truncated Volterra series, the optimal coefficients tend to be very sparse [37]. In [37], the results using the ℓ_0-norm approximation were superior to the results employing the ℓ_1 norm.

In the context of regression as well as in neural networks, it is very common to employ some of these regularizers [7, 10, 18, 40, 41]. Recently, they have also been applied in the context of distributed learning; see [42] and references therein.

In addition to the sparsity-promoting regularizers presented in this chapter, there also exist the so-called *mixed norms* that have been used for block-sparse systems, that is, systems in which the nonzero coefficients appear in a small number of clusters [43, 44]. In this approach, the coefficients are separated in several clusters, but determining an adequate size for these clusters can be challenging. Besides, to the best of our knowledge, this approach has not been tested using real data and compressible signals yet.

2.4 Proportionate-Type Algorithms

In the proportionate-type algorithms, the update term applied to each adaptive filter coefficient $w_i(k)$ is *proportional* to its own magnitude $|w_i(k)|$, meaning that large magnitude coefficients update rapidly (through large steps), whereas small magnitude coefficients update slowly (with small steps). These algorithms are quite interesting in applications involving the identification of systems whose coefficients have a wide range of absolute values, and one has prior information about the value of their small magnitude coefficients, as is the case of a sparse system identification in which we know beforehand that the small coefficients are equal to 0. Such prior information is of paramount importance in these algorithms as they update the small magnitude coefficients very slowly and, therefore, the adaptive filter coefficients must be initialized properly (at least, the coefficients w_i whose optimal values are small in magnitude), as explained in [9].

The proportionate-type algorithms are extensions of the classical algorithms, like the NLMS and AP algorithms, in which this proportionate-update principle

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

is introduced through the so-called *proportionate matrix* $\mathbf{G}(k)$. Matrix $\mathbf{G}(k)$ is diagonal with nonzero entries given by

$$g_i(k) = \mathcal{G}(|w_i(k)|) \text{ , for } i \in \{0, 1, \ldots, N\} \tag{2.31}$$

that is, $g_i(k)$ is a function of $|w_i(k)|$. Essentially, the different proportionate-type algorithms are obtained by making different choices of $g_i(k)$, or equivalently, of this function $\mathcal{G}$.

In this section, we cover some of the most important proportionate-type algorithms. We begin with the simpler ones, which are based on the classical NLMS algorithm, and then we move to more general algorithms based on the SM and affine-projection (AP) ideas. We start with the first algorithm of its kind, the proportionate NLMS (PNLMS) algorithm [45]. Then, we cover the improved PNLMS (IPNLMS) algorithm, the μ-law PNLMS (MPNLMS) algorithm, and the improved MPNLMS (IMPNLMS) algorithm. One should notice that these proportionate-type algorithms have become "less proportional" along the years in order to increase the algorithm robustness, as the results reported for the PNLMS algorithm pointed out several cases in which its performance suffers a significant degradation [9, 46, 47]. Indeed, while in the PNLMS algorithm, we have $\mathcal{G}$ in Equation (2.31) equal to the identity function (in general), in the other algorithms mentioned above, a function $\mathcal{G}$ that grows less quickly with $|w_i(k)|$ is used. We also present the set-membership PNLMS (SM-PNLMS) algorithm.

2.4.1 Proportionate-Type Algorithms Based on the NLMS Recursion

In this subsection, we provide a unified treatment of the proportionate-type algorithms based on the NLMS recursion, herein collectively called proportionate-type NLMS (Pt-NLMS) algorithms, which include the PNLMS, the IPNLMS, the MPNLNS, and the IMPNLMS, among many other algorithms. Specifically, we present their general update equation, the related optimization problem, geometric interpretations, the role of the proportionate matrix $\mathbf{G}(k)$, and we address the choice of the common parameters. Besides, the material presented in this subsection can be adapted to the SM-PNLMS and SM-PAP algorithms easily.

The update equation (recursion) of the proportionate-type algorithms based on the NLMS recursion is given by

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu \frac{e(k)\mathbf{G}(k)\mathbf{x}(k)}{\mathbf{x}^{\mathrm{T}}(k)\mathbf{G}(k)\mathbf{x}(k) + \delta}, \tag{2.32}$$

where $\mathbf{w}(k), \mathbf{x}(k) \in \mathbb{R}^{N+1}$ are the adaptive filter coefficient (weight) vector and the input vector applied to the adaptive filter at iteration k, respectively. Denoting the desired or reference signal (also known as target) by $d(k) \in \mathbb{R}$, the error signal at the k-th iteration is defined as

$$e(k) = d(k) - \mathbf{w}^{\mathrm{T}}(k)\mathbf{x}(k). \tag{2.33}$$

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

The regularization parameter $\delta \in \mathbb{R}_+$ is a small nonnegative number used to avoid numerical issues when $\mathbf{x}^{\mathrm{T}}(k)\mathbf{G}(k)\mathbf{x}(k)$ tends to 0. The step size or learning rate parameter is denoted by $\mu \in \mathbb{R}_+$. Just like in the NLMS algorithm, the step size should be chosen in the range $0 < \mu \leq 1$, and it represents a trade-off between convergence speed and steady-state error, that is, higher (lower) values of μ lead to faster (slower) convergence, but higher (lower) levels of steady-state MSE. Finally, $\mathbf{G}(k)$ is the proportionate matrix, which is a diagonal matrix whose elements on the main diagonal are denoted by $g_i(k), i \in \{0, 1, \ldots, N\}$. As previously explained, the only difference among the proportionate-type algorithms based on the NLMS recursion lies on the definition of this matrix, that is, on the relation between $g_i(k)$ and $|w_i(k)|$ in Equation (2.31), a topic addressed in the subsections to come.

First, observe that the recursion given in Equation (2.32) resembles that of the NLMS algorithm. Indeed, if we choose $\mathbf{G}(k) = \mathbf{I}$ (the identity matrix), then the two recursions coincide. However, unlike the NLMS algorithm, whose update direction is given by $\mathbf{x}(k)$, the update direction in Equation (2.32) is determined by vector $\mathbf{x}'(k) = \mathbf{G}(k)\mathbf{x}(k)$. If $\mathbf{G}(k)$ is chosen properly (i.e., according to the proportional-update principle), then the entries of $\mathbf{x}'(k)$ are amplified or attenuated, in relation to the corresponding entries of $\mathbf{x}(k)$, by a function of the magnitude of its corresponding entry $|w_i(k)|$. The practical effect is that the adaptive filter coefficients $w_i(k)$ with larger (smaller) magnitudes are updated with larger (shorter) steps. The matrix $\mathbf{G}(k)$ appearing in the denominator is responsible for keeping the algorithm normalized.

Second, the recursion given in Equation (2.32) is related to the solution of the following constrained optimization process:

$$\begin{aligned} &\min \|\mathbf{w}(k+1) - \mathbf{w}(k)\|^2_{\mathbf{G}^{-1}(k)} \\ &\text{subject to } d(k) - \mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k+1) = 0 \ , \end{aligned} \tag{2.34}$$

where $\|\mathbf{w}(k+1)-\mathbf{w}(k)\|^2_{\mathbf{G}^{-1}(k)} = [\mathbf{w}(k+1)-\mathbf{w}(k)]^{\mathrm{T}}\mathbf{G}^{-1}(k)[\mathbf{w}(k+1)-\mathbf{w}(k)]$ is a norm induced by the matrix $\mathbf{G}^{-1}(k)$. This is a minimum disturbance problem with linear constraint. The constraint means $\mathbf{w}(k+1)$ must yield an *a posteriori* error equal to 0, that is, it must fit the data $(\mathbf{x}(k), d(k))$ exactly. The cost function forces $\mathbf{w}(k+1)$ to be close to $\mathbf{w}(k)$ in some sense, but not in the usual Euclidean sense, in order to maintain a good fit of the prior data $(\mathbf{x}(i), d(i))$, with $i < k$.

To solve the optimization problem given in Equation (2.34), we apply the method of Lagrange multipliers and form the Lagrangian function

$$\mathcal{L}\left(\mathbf{w}(k+1), \lambda\right) = \|\mathbf{w}(k+1) - \mathbf{w}(k)\|^2_{\mathbf{G}^{-1}(k)} + \lambda\left[d(k) - \mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k+1)\right], \tag{2.35}$$

where $\lambda \in \mathbb{R}$ is the Lagrange multiplier. Then, we differentiate $\mathcal{L}$ with respect to $\mathbf{w}(k+1)$ and λ and equate the results to 0 to obtain

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}(k+1)} = \mathbf{0} \quad \therefore \quad \mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\lambda}{2}\mathbf{G}(k)\mathbf{x}(k), \tag{2.36}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = 0 \quad \therefore \quad \mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k+1) = d(k), \tag{2.37}$$

in which we have assumed that $\mathbf{G}(k)$ exists (i.e., it is nonsingular). If we premultiply Equation (2.36) by $\mathbf{x}^{\mathrm{T}}(k)$, use Equation (2.37), and the definition of the error signal $e(k)$, then we get

$$\frac{\lambda}{2} = \frac{e(k)}{\mathbf{x}^{\mathrm{T}}(k)\mathbf{G}(k)\mathbf{x}(k)}. \tag{2.38}$$

Substituting this relation in Equation (2.36), we obtain the following recursion:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{e(k)\mathbf{G}(k)\mathbf{x}(k)}{\mathbf{x}^{\mathrm{T}}(k)\mathbf{G}(k)\mathbf{x}(k)}. \tag{2.39}$$

Finally, to obtain Equation (2.32), we just need to include two parameters in Equation (2.39): the regularization factor δ in the denominator and the step size parameter μ, also known as *relaxation factor* since, from the optimization point of view, it relaxes the problem constraint by not forcing the *a posteriori* error to be equal to 0 (which happens only when $\mu = 1$).

In this subsection, we addressed some topics common to every proportionate-type algorithm based on the NLMS recursion. In the following subsections, we focus on the main ingredient of the proportionate-type algorithms: the proportionate matrix $\mathbf{G}(k)$.

2.4.2 The PNLMS Algorithm

The PNLMS algorithm was proposed by Duttweiler [45] in 2000, and it inspired the development of many other algorithms following the same proportionate-update principle, but using slightly different proportionate matrices. In [45], the PNLMS algorithm is shown to converge faster than the NLMS algorithm when estimating/identifying impulse responses corresponding to echo paths. Such impulse responses are usually very long (the higher the sampling rate, the longer they are), but most of their energy is concentrated in a few samples, meaning that echo paths are examples of sparse systems found in practice [48].

In the PNLMS recursion given in Equation (2.32), each entry on the main diagonal of the proportionate matrix $\mathbf{G}(k) = \mathsf{Diag}\{[g_0(k)\ g_1(k)\ \cdots\ g_N(k)]\}$ is given by

$$g_i(k) = \frac{\gamma_i(k)}{\sum_{j=0}^{N} |\gamma_j(k)|}, \tag{2.40}$$

$$\gamma_i(k) = \max\left\{ \underbrace{|w_i(k)|}_{1^{\mathrm{st}}\ \mathrm{term}},\ \underbrace{\rho \max\{\delta_\rho,\ |w_0(k)|,\ \cdots,\ |w_N(k)|\}}_{2^{\mathrm{nd}}\ \mathrm{term}} \right\}, \tag{2.41}$$

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

Algorithm 2.7 The PNLMS algorithm

Initialization:

$\mathbf{x}(0) = \mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\mathrm{T}}$

choose μ in the range $0 < \mu \leq 1$

choose ρ in the range $0 < \rho \ll 1$

choose δ and δ_p as small positive constants

For $k \geq 0$ (i.e., for every iteration) **do**

$e(k) = d(k) - \mathbf{w}^{\mathrm{T}}(k)\mathbf{x}(k)$

$\gamma_i(k) = \max\{|w_i(k)|\ ,\ \rho \max\{\delta_p, |w_0(k)|, ..., |w_{\mathrm{N}}(k)|\}\}$, for all i

$$g_i(k) = \frac{\gamma_i(k)}{\sum_{j=0}^{\mathrm{N}} \gamma_j(k)}, \text{ for all } i$$

$\mathbf{G}(k) = \mathsf{Diag}\{[g_0(k)\ g_1(k)\ \ldots\ g_N(k)]\}$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu \frac{e(k)\mathbf{G}(k)\mathbf{x}(k)}{\mathbf{x}^{\mathrm{T}}(k)\mathbf{G}(k)\mathbf{x}(k) + \delta}$$

where $\max\{\cdot\}$ returns the maximum among the elements of a given set, $\delta_\rho \in \mathbb{R}_+$ is used to prevent $\mathbf{w}(k)$ from stalling during the initialization stage, in case the coefficients are initialized as $\mathbf{w}(0) = \mathbf{0}$ and its usual value is $\delta_\rho = 0.01$, and $\rho \in \mathbb{R}_+$ is used to prioritize the first term in Equation (2.41) and its typical value is $\rho = 0.01$. However, we should emphasize that the proper choices of δ_ρ and ρ depend on prior knowledge about the application (more precisely, about the magnitudes of the optimal coefficients). The complete description of the PNLMS algorithm is given in Algorithm 2.7.

Remarks:

1 The first term in Equation (2.41) conveys the proportional-update idea, but the second term is necessary to address the cases in which $w_i(k) = 0$. That is, if we were to use only the first term, then the update process of a given coefficient $w_i(k)$ would stall whenever $w_i(k) = 0$, thus hindering the initialization and also the tracking of time-varying systems.

2 To better understand the proportionate-update principle, let us consider just the first term in Equation (2.41), which leads to

$$g_i(k) = \frac{|w_i(k)|}{\sum_{j=0}^{N} |w_j(k)|} = \frac{|w_i(k)|}{\|\mathbf{w}(k)\|_1}, \forall i, \tag{2.42}$$

where $\|\cdot\|_1$ denotes the ℓ_1 norm. In addition, observe that the denominator of Equation (2.42) does not change with i and, therefore, it vanishes when Equation (2.42) is substituted in Equation (2.39) due to the presence of matrix $\mathbf{G}(k)$ on both the numerator and denominator of this equation. Consequently, we can think of Equation (2.42) as $g_i(k) \propto |w_i(k)|$

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

(proportional relation), which means that the update step applied to each coefficient $w_i(k)$ is proportional to its own magnitude. Moreover, since

$$\sum_{i=0}^{N} g_i(k) = 1, \tag{2.43}$$

the product $\mu g_i(k)$ can be interpreted as an "equivalent step size" for the i-th coefficient, that is, the fraction of μ corresponding to the update of $w_i(k)$. This interpretation allows us to conclude that in the proportionate-update framework, the step size μ is unevenly distributed across the coefficients, privileging those with higher magnitudes, but impairing the lower magnitude ones.

3 If the filter coefficients are initialized as $\mathbf{w}(0) = \mathbf{0}$, then $\gamma_i(0) = \rho\delta_\rho$ for all i, meaning that

$$g_i(k) = \frac{1}{N+1}, \ \forall i, \tag{2.44}$$

which means that the step size is evenly distributed across the coefficients, just like in the NLMS algorithm.

4 The PNLMS algorithm works well when the impulse response of the system to be identified is very sparse, resembling the Kronecker's delta function. On the other hand, as the sparsity degree decreases, the performance of the PNLMS algorithm deteriorates, in comparison with the standard algorithms.

2.4.3 The IPNLMS Algorithm

The IPNLMS algorithm proposed in 2002 [46] is more robust to lower sparsity degrees than the PNLMS algorithm, providing good results even when applied to dispersive systems. The IPNLMS algorithm benefits from the combination of the proportional update term with the standard update term of the NLMS algorithm, which works better than the PNLMS algorithm in the estimation of dispersive systems.

In the IPNLMS recursion given in Equation (2.32), each entry on the main diagonal of the proportionate matrix $\mathbf{G}(k) = \mathsf{Diag}\{[g_0(k)\ g_1(k)\ \cdots\ g_N(k)]\}$ is given by

$$g_i(k) = \underbrace{\frac{1-\alpha}{2(N+1)}}_{\text{NLMS term}} + \underbrace{\frac{(1+\alpha)|w_i(k)|}{2\|\mathbf{w}(k)\|_1 + \delta}}_{\text{PNLMS term}}, \tag{2.45}$$

where $\delta \in \mathbb{R}_+$ is also a regularization factor used to avoid numerical problems when $\|\mathbf{w}(k)\|_1$ tends to 0 and $\alpha \in \mathbb{R}$ is an adjustable parameter that represents a tradeoff between the NLMS and PNLMS terms in Equation (2.45). In fact, α should be chosen in the range $-1 \leq \alpha < 1$. If $\alpha = -1$, then the IPNLMS is equivalent to the NLMS algorithm, whereas for $\alpha \approx 1$, its update resembles that of the PNLMS algorithm. Typically, this parameter is chosen close to $\alpha = -0.5$

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

Algorithm 2.8 The IPNLMS algorithm

Initialization:

$\mathbf{x}(0) = \mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\mathrm{T}}$

choose μ in the range $0 < \mu \leq 1$

choose δ as a small positive constant

choose α in the range $-1 \leq \alpha < 1$

For $k \geq 0$ (i.e., for every iteration) **do**

$$e(k) = d(k) - \mathbf{w}^{\mathrm{T}}(k)\mathbf{x}(k)$$

$$g_i(k) = \frac{1-\alpha}{2(N+1)} + \frac{(1+\alpha)|w_i(k)|}{2\|\mathbf{w}(k)\|_1 + \delta}$$

$$\mathbf{G}(k) = \mathsf{Diag}\{[g_0(k)\ g_1(k)\ \ldots\ g_N(k)]\}$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu \frac{e(k)\mathbf{G}(k)\mathbf{x}(k)}{\mathbf{x}^{\mathrm{T}}(k)\mathbf{G}(k)\mathbf{x}(k) + \delta}$$

so that the IPNLMS algorithm behaves more like an NLMS than a PNLMS algorithm. It is interesting to observe that the NLMS term in Equation (2.45) already prevents the IPNLMS coefficients from stalling, which is another reason for not allowing $\alpha = 1$. The IPNLMS algorithm is given in Algorithm 2.8.

2.4.4 The MPNLMS Algorithm

When applied to the identification of sparse impulse responses, the PNLMS algorithm usually provides a very fast convergence speed in the early iterations, but later it slows down. The MPNLMS algorithm proposed in 2005 [47] addresses this issue. The key idea of the MPNLMS algorithm is to employ step sizes proportional to a function of the magnitude of the coefficients; such function must grow rapidly for low magnitude coefficients, increasing the resolution in this range, but must grow slowly for high magnitude coefficients, thus preventing numerical issues [47].

In the MPNLMS recursion given in Equation (2.32), each entry on the main diagonal of the proportionate matrix $\mathbf{G}(k) = \mathsf{Diag}\{[g_0(k)\ g_1(k)\ \cdots\ g_N(k)]\}$ is given by

$$g_i(k) = \frac{\hat{\gamma}_i(k)}{\sum_{j=0}^{N} |\hat{\gamma}_j(k)|}, \tag{2.46}$$

$$\hat{\gamma}_i(k) = \max\left\{F(|w_i(k)|),\ \rho \max\left\{\delta_\rho,\ F(|w_0(k)|),\ \cdots,\ F(|w_N(k)|)\right\}\right\}, \tag{2.47}$$

$$F(|w_i(k)|) = \ln\left(1 + \mu_F |w_i(k)|\right), \tag{2.48}$$

where $\delta_\rho \in \mathbb{R}_+$ is used to prevent $\mathbf{w}(k)$ from stalling during the initialization stage in case the coefficients are set as $\mathbf{w}(0) = \mathbf{0}$ and its typical value is $\delta_\rho = 0.01$, $\rho \in \mathbb{R}_+$ is used to prioritize the term $F(|w_i(k)|)$ in Equation (2.47) and its typical value is $\rho = 0.01$, and $\mu_F \in \mathbb{R}_+$ is a large positive number

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

Algorithm 2.9 The MPNLMS algorithm

Initialization:

$\mathbf{x}(0) = \mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\mathrm{T}}$

choose μ in the range $0 < \mu \leq 1$

choose ρ in the range $0 < \rho \ll 1$

choose δ and δ_p as small positive constants

choose μ_F as a large positive number

For $k \geq 0$ (i.e., for every iteration) **do**

$e(k) = d(k) - \mathbf{w}^{\mathrm{T}}(k)\mathbf{x}(k)$

$F(|w_i(k)|) = \ln(1 + \mu_F|w_i(k)|)$, for all i

$\hat{\gamma}_i(k) = \max\{F(|w_i(k)|),\ \rho\max\{\delta_\rho,\ F(|w_0(k)|),\ \cdots,\ F(|w_N(k)|)\}\}$, $\forall i$

$$g_i(k) = \frac{\hat{\gamma}_i(k)}{\sum_{j=0}^{N}|\hat{\gamma}_j(k)|}, \text{ for all } i$$

$\mathbf{G}(k) = \mathsf{Diag}\{[g_0(k)\ g_1(k)\ \ldots\ g_N(k)]\}$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu\frac{e(k)\mathbf{G}(k)\mathbf{x}(k)}{\mathbf{x}^{\mathrm{T}}(k)\mathbf{G}(k)\mathbf{x}(k) + \delta}$$

related to the identification accuracy requirement, which is typically chosen as $\mu_F = 1000$. Observe that the main difference between the MPNLMS and PNLMS algorithms is the use of a natural logarithmic function by the former one to achieve the benefits mentioned in the previous paragraph. The complete description of the MPNLMS algorithm is given in Algorithm 2.9.

Remarks:

1 As in the PNLMS algorithm, proper selection of the parameters ρ, δ_ρ, and μ_F require some prior knowledge about the application/impulse response to be estimated.

2 If the filter coefficients are initialized as $\mathbf{w}(0) = \mathbf{0}$, then $\hat{\gamma}_i(0) = \rho\delta_\rho$ for all i, meaning that

$$g_i(k) = \frac{1}{N+1}, \forall i,$$

and the MPNLMS algorithm will act just like an NLMS algorithm.

3 In normal operation, $\hat{\gamma}_i(k)$ should be equal to the first term in Equation (2.47) leading to

$$g_i(k) = \frac{F(|w_i(k)|)}{\sum_{j=0}^{N}F(|w_j(k)|)},$$

that is, in the MPNLMS algorithm, the step sizes applied to each coefficient are proportional to $F(|w_i(k)|)$, and not $|w_i(k)|$ as in the PNLMS algorithm.

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

2.4.5 The IMPNLMS Algorithm

Just like the PNLMS, the MPNLMS algorithm also has poor performance when identifying systems that are not very sparse. A natural idea to improve this algorithm is to follow the same approach used in the IPNLMS algorithm, that is, to use "equivalent step sizes" formed by the combination of two terms: one corresponding to the NLMS and the other corresponding to the MPNLMS algorithm. In addition to adapting the IPNLMS idea, the IMPNLMS algorithm proposed in 2008 [49] also employs a measure of the sparsity degree in the coefficient vector $\mathbf{w}(k)$ in order to select the parameter $\alpha(k)$ automatically (the parameter that determines the weight given to each of the two terms above).

In the IMPNLMS recursion given in Equation (2.32), each entry on the main diagonal of the proportionate matrix $\mathbf{G}(k) = \mathsf{Diag}\{[g_0(k)\ g_1(k)\ \cdots\ g_N(k)]\}$ is given by

$$g_i(k) = \underbrace{\frac{1-\alpha(k)}{2(N+1)}}_{\text{NLMS term}} + \underbrace{\frac{(1+\alpha(k))F(|w_i(k)|)}{2\|F(|\mathbf{w}(k)|)\|_1+\delta}}_{\text{MPNLMS term}}, \tag{2.49}$$

where

$$F(|w_i(k)|) = \ln\left(1+\mu_F|w_i(k)|\right), \tag{2.50}$$

$$\alpha(k) = 2\xi(k) - 1, \tag{2.51}$$

$$\xi(k) = (1-\lambda)\xi(k-1) + \lambda\xi_{\mathbf{w}}(k), \tag{2.52}$$

$$\xi_{\mathbf{w}}(k) = \frac{N+1}{N+1-\sqrt{N+1}}\left(1 - \frac{\|\mathbf{w}(k)\|_1}{\sqrt{N+1}\|\mathbf{w}(k)\|_2}\right), \tag{2.53}$$

$F(|\mathbf{w}(k)|) = [F(|w_0(k)|)\ F(|w_1(k)|)\ \ldots\ F(|w_N(k)|)]^{\mathrm{T}} \in \mathbb{R}_+^{N+1}$, $\delta \in \mathbb{R}_+$ is a regularization factor, $\mu_F \in \mathbb{R}_+$ is a large positive number related to the identification accuracy requirement and is typically chosen as $\mu_F = 1000$, and $\lambda \in \mathbb{R}_+$ should be chosen as $0 < \lambda \ll 1$. Low values of λ privilege the past data, whereas high values of λ prioritize the instantaneous estimate of the sparsity degree given by $\xi_{\mathbf{w}}(k)$. Observe that if $\xi(k) \approx 1$, corresponding to a vector $\mathbf{w}(k)$ with high sparsity degree, then $\alpha(k) \approx 1$, meaning that the NLMS term in $g_i(k)$ vanishes and, consequently, the algorithm acts like the MPNLMS algorithm. On the contrary, if $\xi(k) \approx 0$, corresponding to a dispersive vector $\mathbf{w}(k)$, then $\alpha(k) \approx -1$ and the MPNLMS term vanishes.

The complete description of the IMPNLMS algorithm is given in Algorithm 2.10.

2.4.6 The SM-PNLMS Algorithm

The SM-PNLMS algorithm proposed in [50] combines the proportionate-update principle with the SM filtering concept, which allows the adaptive filter to update its coefficients only when the input data brings enough innovation [30, 60].

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

Algorithm 2.10 The IMPNLMS algorithm

Initialization:

$\mathbf{x}(0) = \mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\mathrm{T}}$

$\xi(0) = 0$

choose μ in the range $0 < \mu \leq 1$

choose δ as a small positive number

choose μ_F as a large positive number

choose λ in the range $0 < \lambda \ll 1$

For $k \geq 0$ (i.e., for every iteration) **do**

$$e(k) = d(k) - \mathbf{w}^{\mathrm{T}}(k)\mathbf{x}(k)$$

$$\xi_{\mathbf{w}}(k) = \frac{N+1}{N+1-\sqrt{N+1}}\left(1 - \frac{\|\mathbf{w}(k)\|_1}{\sqrt{N+1}\|\mathbf{w}(k)\|_2}\right)$$

$$\xi(k) = (1-\lambda)\xi(k-1) + \lambda\xi_{\mathbf{w}}(k)$$

$$\alpha(k) = 2\xi(k) - 1$$

$$F(|w_i(k)|) = \ln(1 + \mu_F|w_i(k)|), \text{ for all } i$$

$$g_i(k) = \frac{1-\alpha(k)}{2(N+1)} + \frac{(1+\alpha(k))F(|w_i(k)|)}{2\|F(|\mathbf{w}(k)|)\|_1 + \delta}, \text{ for all } i$$

$$\mathbf{G}(k) = \mathsf{Diag}\{[g_0(k)\ g_1(k)\ \ldots\ g_N(k)]\}$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu\frac{e(k)\mathbf{G}(k)\mathbf{x}(k)}{\mathbf{x}^{\mathrm{T}}(k)\mathbf{G}(k)\mathbf{x}(k) + \delta}$$

The update equation of the SM-PNLMS algorithm is slightly different from that in Equation (2.32), as the former uses a time-varying step-size parameter $\mu(k)$, and can be written as

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu(k)\frac{e(k)\mathbf{G}(k)\mathbf{x}(k)}{\mathbf{x}^{\mathrm{T}}(k)\mathbf{G}(k)\mathbf{x}(k) + \delta}, \tag{2.54}$$

where $\mathbf{G}(k) = \mathsf{Diag}\{[g_0(k)\ \cdots\ g_N(k)]\}$,

$$g_i(k) = \frac{1-\kappa\mu(k)}{N+1} + \frac{\kappa\mu(k)|w_i(k)|}{\|\mathbf{w}(k)\|_1 + \delta}, \tag{2.55}$$

$$\mu(k) = \begin{cases} 1 - \frac{\overline{\gamma}}{|e(k)|} & \text{if } |e(k)| > \overline{\gamma} \\ 0 & \text{otherwise.} \end{cases} \tag{2.56}$$

Thus, the update equation can be rewritten in a more direct form as follows:

$$\mathbf{w}(k+1) = \begin{cases} \mathbf{w}(k) + \left(1 - \frac{\overline{\gamma}}{|e(k)|}\right)\frac{e(k)\mathbf{G}(k)\mathbf{x}(k)}{\mathbf{x}^{\mathrm{T}}(k)\mathbf{G}(k)\mathbf{x}(k) + \delta} & \text{if } |e(k)| > \overline{\gamma}, \\ \mathbf{w}(k) & \text{otherwise.} \end{cases} \tag{2.57}$$

where $\delta \in \mathbb{R}_+$ is a *regularization factor* used to prevent divisions by 0, $\overline{\gamma} \in \mathbb{R}_+$ is the prescribed threshold that defines how much residual error is acceptable, and $\kappa \in [0, 1)$ represents a compromise between the NLMS update ($\kappa \to 0$) and the proportionate update ($\kappa \to 1$) and is usually chosen as $\kappa = 0.5$. Observe

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

Algorithm 2.11 The SM-PNLMS algorithm

Initialization:

$\mathbf{x}(0) = \mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\mathrm{T}}$

choose δ as a small positive constant

choose κ in the range $0 \leq \kappa < 1$

For $k \geq 0$ (i.e., for every iteration) **do**

$e(k) = d(k) - \mathbf{w}^{\mathrm{T}}(k)\mathbf{x}(k)$

If $|e(k)| > \overline{\gamma}$

$\mu(k) = 1 - \frac{\overline{\gamma}}{|e(k)|}$

$g_i(k) = \dfrac{1-\kappa\mu(k)}{N+1} + \dfrac{\kappa\mu(k)|w_i(k)|}{\|\mathbf{w}(k)\|_1 + \delta}$, for all i

$\mathbf{G}(k) = \mathsf{Diag}\{[g_0(k)\ g_1(k)\ \ldots\ g_N(k)]\}$

$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu(k)\dfrac{e(k)\mathbf{G}(k)\mathbf{x}(k)}{\mathbf{x}^{\mathrm{T}}(k)\mathbf{G}(k)\mathbf{x}(k) + \delta}$

Else

$\mathbf{w}(k+1) = \mathbf{w}(k)$

End if

that the choice of $\overline{\gamma}$ depends on some previous knowledge about the problem uncertainties. For applications in which the desired signal is corrupted by an additive noise $n(k)$ with variance denoted by σ_n^2, a common choice is to select $\overline{\gamma} = \sqrt{5\sigma_n^2}$ [51].

Notice that the elements $g_i(k)$ in the SM-PNLMS algorithm are very similar to the ones used in the IPNLMS algorithm. Consequently, the SM-PNLMS algorithm's performance is usually similar to that of the IPNLMS algorithm, but the former algorithm has two advantages: (i) it does not update at every iteration, thus saving computational resources and (ii) it uses a variable step-size $\mu(k)$ which is automatically chosen. The full description of the SM-PNLMS algorithm is given in Algorithm 2.11.

2.4.7 Numerical Experiments

Here, the PNLMS, IPNLMS, and MPNLMS algorithms are used to identify an unknown system whose coefficients are denoted by $\mathbf{w}_{\mathrm{o}}$. We also consider two other algorithms: the NLMS, representing the baseline for our comparisons, and the ℓ_0-NLMS, representing an algorithm that uses a sparsity-promoting regularizer.

The specific parameters of these Pt-NLMS algorithms were set according to their typical values, described in the subsections where these algorithms were explained. As for the ℓ_0-NLMS algorithm, we used the Laplacian function with $\beta = 5$. For all algorithms, we set $\delta = 10^{-12}$, $\alpha = 2 \times 10^{-3}$, and the step size μ is informed later. The input signal is a zero-mean white Gaussian noise (WGN) with unitary variance. The measurement noise is also zero-mean WGN with variance $\sigma_n^2 = 10^{-2}$ and is uncorrelated with the input signal. We assume the

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

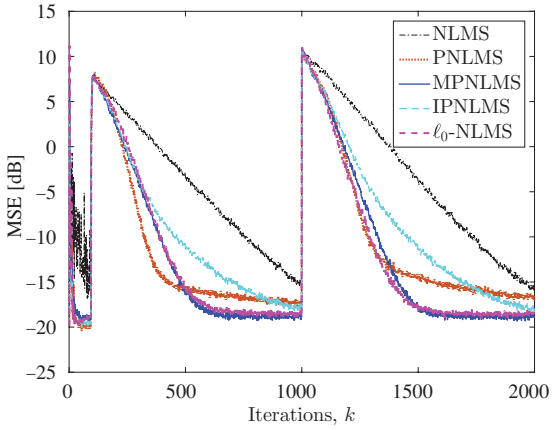


Figure 2.6 MSE learning curve in a nonstationary scenario.

adaptive filter and the unknown systems have the same number of coefficients. The MSE learning curves for each algorithm are generated by averaging the outcomes of 1000 independent trials.

In Figure 2.6, we consider a nonstationary scenario in which the optimal coefficients are given by $\mathbf{w}_{\mathrm{o}}^{(1)}$ during the first 1000 iterations, and then they change to $\mathbf{w}_{\mathrm{o}}^{(2)}$, defined as

$$\mathbf{w}_{\mathrm{o},i}^{(1)}=\begin{cases}0, & \text{for } 0\leq i\leq 93,\\ 1, & \text{for } 94\leq i\leq 99,\end{cases}\qquad \mathbf{w}_{\mathrm{o},i}^{(2)}=\begin{cases}1, & \text{for } 0\leq i\leq 5,\\ 0, & \text{for } 6\leq i\leq 99.\end{cases} \tag{2.58}$$

Both $\mathbf{w}_{\mathrm{o}}^{(1)}$ and $\mathbf{w}_{\mathrm{o}}^{(2)}$ have 6 nonzero coefficients out of 100. The algorithms were initialized with $\mathbf{w}(0)=\mathbf{0}$, and the step sizes of the NLMS, PNLMS, IPNLMS, MPNLNS and ℓ_0-NLMS algorithms were set as 0.4, 0.15, 0.4, 0.6, and 0.99, respectively, so that they could achieve the same steady-state MSE level as fast as possible. In this figure, both the MPNLMS and ℓ_0-NLMS algorithms were the fastest ones to reach a specific MSE level; actually, the ℓ_0-NLMS algorithm was a bit faster in the tracking of $\mathbf{w}_{\mathrm{o}}^{(2)}$. The PNLMS algorithm started very fast but then its convergence speed slowed down at a given iteration; this behavior is typical since the PNLMS algorithm accelerates the convergence of the high-magnitude coefficients in detriment of the low-magnitude ones. Finally, the results of the IPNLMS algorithm correspond to a balance between the PNLMS and NLMS algorithms.

2.4.8 Further Readings

The literature about proportionate-type algorithms is vast. For different versions of Pt-NLMS algorithms, one may refer to [52–55]. Analogously, for several versions of proportionate AP algorithms, one may refer to [56–58]. Recently,

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

an approach called *coefficient vector reusing* has received some attention [59]. In the same way that the AP algorithm generalizes the NLMS algorithm by reusing previous input vectors, the SM-PAP algorithm proposed in [60] generalizes the SM-PNLMS algorithm.

2.5 Conclusion

This chapter presented many adaptive filtering algorithms incorporating tools to exploit the sparsity inherent to some models in real applications. The chapter started by covering in some detail the sparsity-promoting regularizations frequently used to exploit sparsity. Combined with the classical adaptive filtering algorithms, these regularizations form a set of solutions giving rise to several sparsity-aware online algorithms. Another class of algorithms that can exploit sparsity is the proportionate-type adaptive filtering algorithms, in which coefficient updates are proportional, to a certain extent (depending on the algorithm), to their own magnitudes. When comparing these two classes of algorithms, one can observe how different their principles are. While the algorithms employing sparsity-promoting regularizations push some coefficients to 0 in order to encourage sparse solutions (in the case of the ℓ_0-norm approximation, for example, only the low magnitude coefficients suffer from this zero-attraction effect, whereas the relevant coefficients remain unaltered), the proportionate-type algorithms act by realizing an uneven distribution of the step size μ among the coefficients, which accelerates the convergence of high magnitude coefficients, but slows down the convergence of low magnitude coefficients. That is, the algorithms employing sparsity-promoting regularizations focus on the low magnitude coefficients, whereas the proportionate-type algorithms focus on the high magnitude coefficients.

From our experience with the algorithms described in this chapter, we can state that:

1 Considering the algorithms employing sparsity-promoting regularizers, the ℓ_0-norm approximation has always yielded the best results in our tests, and it is only slightly more complex than the ℓ_1 norm.
2 As for the algorithms following the proportional-update principle, the ones whose recursions are "more proportional" to the magnitude of the coefficients usually work very well in highly sparse scenarios, but their performance degrade severely as the sparsity degree decreases. In our tests, the MPNLMS and SM-PNLMS algorithms are usually more robust to different simulation setups than the other Pt-NLMS algorithms.
3 Another advantage of algorithms employing the ℓ_0-norm approximation is that they always capitalize on sparse vectors, even when the sparsity degree

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

is low (confer the MSE analysis of the ℓ_0-AP algorithm in [28]). The same behavior is not observed in other regularizers nor in the proportionate-type algorithms. Besides, parameter β provides an easy mechanism to adjust the ℓ_0-norm approximation to deal with compressible vectors.

Finally, we should say that the online learning algorithms that exploit sparse models presented in this chapter are far from representing the entire set of solutions. Some examples of alternative approaches and applications can be found in [61–66]. Also, in [67], an algorithm combining both the sparsity-promoting regularization and the proportional-update principle has shown some interesting results.

Problems

2.1 For under-determined problems of the type $\mathbf{A}\mathbf{x} = \mathbf{b}$, it is a common practice to include some constraint on $\mathbf{x}$ in order to find a single solution. Considering the data:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 0 & 2 \\ 1 & 2 & 2 & 0 \\ 1 & 3 & 0 & 0 \end{bmatrix} \text{ and } \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}$$

(a) Compute the minimum ℓ_2 (Euclidean) norm solution $\hat{\mathbf{x}}_2 = \mathbf{A}^{\dagger}\mathbf{b}$, where $\mathbf{A}^{\dagger} = \mathbf{A}^{\mathrm{T}}\left(\mathbf{A}\mathbf{A}^{\mathrm{T}}\right)^{-1}$ is known as pseudo-inverse (or right inverse of $\mathbf{A}$).
(b) Compute the minimum ℓ_1-norm solution $\hat{\mathbf{x}}_1$.
(c) Compute the minimum ℓ_0-norm (sparsest) solution $\hat{\mathbf{x}}_0$. Is there a closed-form solution? Is this solution unique?
Hint: Since $\mathbf{x} \in \mathbb{R}^4$ lives in a low-dimensional space, you can write a program to test all possible combinations of the columns of $\mathbf{A}$. This is called brute-force solution or exhaustive search. Observe that for $\mathbf{A} \in \mathbb{R}^{m \times n}$, this procedure leads to $\sum_{p=1}^{n} \binom{n}{p}$ possibilities that need to be tested. Since large values of n are frequently used in practical applications, the exhaustive search solution is not feasible.

2.2 Prove that the $r\ell_1$ norm, defined in Equation (2.5), converges to the ℓ_0 norm as $\epsilon \to \infty$ and converges to the ℓ_1 norm as $\epsilon \to 0$.

2.3 For $\mu = 1$ and $\gamma = 0$, show that the recursion of the sparsity-aware AP algorithms, given in Equation (2.26), leads to a $\mathbf{w}(k+1)$ that satisfies the constraint in Equation (2.20). Also, show that the corresponding qAP algorithms, whose general recursion is given in Equation (2.27), do not satisfy the constraint in Equation (2.20).

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

2.4 The PAP algorithm is characterized by the recursion

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu \mathbf{G}(k)\mathbf{X}(k)\mathbf{S}(k)\mathbf{e}(k),$$

where $\mathbf{S}(k) = \left(\mathbf{X}^{\mathrm{T}}(k)\mathbf{G}(k)\mathbf{X}(k) + \delta \mathbf{I}\right)^{-1}$. Show that this PAP recursion with $\mu = 1$ and $\delta = 0$ is obtained as the solution of the following optimization problem:

$$\begin{aligned} &\min \|\mathbf{w}(k+1) - \mathbf{w}(k)\|^2_{\mathbf{G}^{-1}(k)} \\ &\text{subject to } \mathbf{d}(k) - \mathbf{X}^{\mathrm{T}}(k)\mathbf{w}(k+1) = \mathbf{0}\ . \end{aligned}$$

2.5 Consider the problem of identifying a highly sparse unknown system/vector $\mathbf{w}_{\mathrm{o}}$ comprised of 100 coefficients. Assume that the input signal is a zero-mean WGN with unitary variance, the measurement noise is also a zero-mean WGN with variance 0.01, and these signals are uncorrelated with each other. Also, the adaptive filter is comprised of 100 coefficients initialized as $\mathbf{w}(0) = \mathbf{0}$. Plot and compare the MSE learning curves for the NLMS, PNLMS, IPNLMS, MPNLMS, IMPNLMS, and ℓ_0-NLMS algorithms considering that:
(a) Only the first coefficient of $\mathbf{w}_{\mathrm{o}}$ is non-null and equal to 1.
(b) Only the 50th coefficient of $\mathbf{w}_{\mathrm{o}}$ is non-null and equal to 1.
(c) Only the last coefficient of $\mathbf{w}_{\mathrm{o}}$ is non-null and equal to 1.
Observe that the sparsity degree is precisely the same in all these cases. So, why the MSE results change in each of them?
Hint: In order to compare the algorithms in a fair manner, adjust their parameters so that they have the same convergence/learning rate and compare the steady-state MSE level they achieve. Do not forget to report the parameters used in each algorithm!

2.6 Repeat the item (a) of Problem 2.5, but for different initialization of the adaptive filter coefficients: $\mathbf{w}(0) = c \times \mathbf{1}$, where $\mathbf{1}$ is the vector of ones and $c \in \{0, 0.1, 0.5, 2\}$. Explain why some of these algorithms suffer a significant performance degradation as c increases.

2.7 Repeat the Problem 2.5, but considering that $\mathbf{w}_{\mathrm{o}}$ is a compressible vector. To allow some control over the "low magnitude" coefficients, take the vectors $\mathbf{w}_{\mathrm{o}}$ given in Problem 2.5 and replace their null coefficients with $p \in \{0.001, 0.005, 0.01, 0.02\}$. Verify which algorithms are mostly impaired by the increase of p.

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

2.8 Repeat the Problem 2.5, but considering that $\mathbf{w}_o$ is a compressible vector. To do so, the entries of $\mathbf{w}_o$ that are equal to 0 must be replaced by a perturbation modeled by a uniform distribution over the interval $[-p, p]$, with $p \in \{0.001, 0.005, 0.01, 0.05\}$. Verify which algorithms are mostly impaired by the increase of p.

2.9 Repeat the Problem 2.5, but varying the degree of sparsity in $\mathbf{w}_o$ by making:

(a) Only the first coefficient of $\mathbf{w}_o$ is non-null and equal to 1.
(b) The first 10 coefficients of $\mathbf{w}_o$ are non-null and equal to 1.
(c) The first half of the coefficients of $\mathbf{w}_o$ are non-null and equal to 1.
(d) All the 100 coefficients of $\mathbf{w}_o$ are non-null and equal to 1.

Verify which algorithms are mostly impaired by the reduction of the sparsity degree.

References

[1] P. S. R. Diniz, H. Yazdanpanah, and M. V. S. Lima, Feature LMS algorithms. Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2018), Calgary, AB, 2018, pp. 4144–4148.

[2] H. Yazdanpanah, P. S. R. Diniz, and M. V. S. Lima, Feature adaptive filtering: Exploiting hidden sparsity, IEEE Transactions on Circuits and Systems I: Regular Papers **67**, pp. 2358–2371 (2020).

[3] H. Yazdanpanah and J. A. Apolinário Jr., The extended feature LMS algorithm: Exploiting hidden sparsity for systems with unknown spectrum, Circuits, Systems, and Signal Processing **40**, pp. 174–192 (2021).

[4] D. B. Haddad, L. O. dos Santos, L. F. Almeida, G. A. S. Santos, and M. R. Petraglia, ℓ_2-norm feature least mean square algorithm, Electronics Letters **56**, pp. 516–519 (2020).

[5] P. S. R. Diniz, H. Yazdanpanah, and M. V. S. Lima, Feature LMS algorithm for bandpass system models. Proceedings of the 27th European Signal Processing Conference (EUSIPCO), A Coruna, Spain, 2019, pp. 1–5.

[6] H. Yazdanpanah, P. S. R. Diniz, and M. V. S. Lima, Low-complexity feature stochastic gradient algorithm for block-lowpass systems, IEEE Access **7**, 141587–141593 (2019).

[7] C. M. Bishop, *Pattern Recognition and Machine Learning* (Springer, New York, 2006).

[8] L. N. Trefethen and D. Bau, III, *Numerical Linear Algebra* (SIAM, Philadelphia, 1997).

[9] M. V. S. Lima, G. S. Chaves, T. N. Ferreira, and P. S. R. Diniz, Do proportionate algorithms exploit sparsity?, ArXiv: http://arxiv.org/abs/2108.06846.

[10] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. (Springer, New York, 2017).

https://doi.org/10.1017/9781108896139.003 Published online by Cambridge University Press

[11] 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).
[12] Y. Eldar and G. Kutyniok, *Compressed Sensing: Theory and Applications* (Cambridge University Press, Cambridge, 2012).
[13] N. Bourbaki, *Topological Vector Spaces* (Springer, New York, 1987).
[14] G. Kothe, *Topological Vector Spaces I* (Springer, New York, 1983).
[15] J. F. Claerbout and F. Muir, Robust modeling with erratic data, Geophysics **38**, pp. 826–844 (1973).
[16] F. Santosa and W. W. Symes, Linear inversion of band-limited reflection seismograms, SIAM Journal on Scientific Computing **7**, pp. 1307–1330 (1986).
[17] S. S. Chen, D. L. Donoho, and M. A. Saunders, Atomic decomposition by basis pursuit, SIAM Journal on Scientific Computing **20**, pp. 33–61 (1998).
[18] R. Tibshirani, Regression shrinkage and selection via the Lasso, Journal of the Royal Statistical Society, Series B (Methodological) **58**, pp. 267–288 (1996).
[19] E. J. Candès and M. B. Wakin, An introduction to compressive sampling, IEEE Signal Processing Magazine **25**, pp. 21–30 (2008).
[20] E. J. Candès, M. B. Wakin, and S. P. Boyd, Enhancing sparsity by reweighted ℓ_1 minimization, Journal of Fourier Analysis and Applications **14**, pp. 877–905 (2008).
[21] M. V. S. Lima, W. A. Martins, and P. S. R. Diniz, Affine projection algorithms for sparse system identification. Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2013), Vancouver, Canada, May 2013, pp. 5666–5670.
[22] L. Mancera and J. Portilla, L0-norm-based sparse representation through alternate projections. Proceedings of the IEEE International Conference on Image Processing (ICIP 2006), Atlanta, GA, USA, October 2006, pp. 2089–2092.
[23] Y. Chen, Y. Gu, and A. O. Hero, Sparse LMS for system identification. Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2009), Taipei, Taiwan, April 2009, pp. 3125–3128.
[24] J. Trzasko and A. Manduca, Highly undersampled magnetic resonance image reconstruction via homotopic ℓ_0-minimization, IEEE Transactions on Medical Imaging **28**, 106–121 (2009).
[25] Y. Gu, J. Jin, and S. Mei, ℓ_0 norm constraint LMS algorithm for sparse system identification, IEEE Signal Processing Letters **16**, pp. 774–777 (2009).
[26] P. Huber, *Robust Statistics* (Wiley, New York, 1981).
[27] D. Geman and G. Reynolds, Nonlinear image recovery with half-quadratic regularization, IEEE Transactions on Image Processing **4**, 932–946 (1995).
[28] M. V. S. Lima, I. Sobron, W. A. Martins, and P. S. R. Diniz, Stability and MSE analyses of affine projection algorithms for sparse system identification. Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2014), Florence, Italy, May 2014, pp. 6399–6403.
[29] H. Mohimani, M. Babaie-Zadeh, and C. Jutten, A fast approach for overcomplete sparse decomposition based on smoothed ℓ^0 norm, IEEE Transactions on Signal Processing **57**, 289–301 (2009).
[30] P. S. R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*, 5th ed. (Springer, New York, 2020).

[31] R. Meng, R. C. de Lamare, and V. H. Nascimento, Sparsity-aware affine projection adaptive algorithms for system identification. Proceedings of the Sensor Signal Processing for Defense (SSPD 2011), London, UK, September 2011, pp. 1–5.

[32] H. Yazdanpanah and P. S. R. Diniz, Recursive least-squares algorithms for sparse system modeling. Proceedings of the 2017 IEEE International Conference on Acoustics Speech and Signal Processing, New Orleans, LA, March 2017, pp. 3878–3883.

[33] H. Yazdanpanah, M. V. S. Lima, and P. S. R. Diniz, On the robustness of set-membership adaptive filtering algorithms, EURASIP Journal on Advances in Signal Processing **2017**, pp. 1–12 (2017).

[34] P. L. Combettes, The foundations of set theoretic estimation, Proceedings of the IEEE **81**, pp. 182–208 (1993).

[35] M. V. S. Lima and P. S. R. Diniz, Fast learning set theoretic estimation. Proceedings of the 21st European Signal Processing Conference (EUSIPCO), Marrakesh, Morocco, 2013, pp. 1–5.

[36] H. Yazdanpanah, J. A. Apolinário Jr., P. S. R. Diniz, and M. V. S. Lima, ℓ_0-norm feature LMS algorithms. Proceedings of the IEEE Global Conference on Signal and Information Processing (GlobalSIP), Anaheim, CA, USA, 2018, pp. 311–315.

[37] H. Yazdanpanah, A. Carini, and M. V. S. Lima, L_0-norm adaptive Volterra filters. Proceedings of the 27th European Signal Processing Conference (EUSIPCO), A Coruna, Spain, 2019, pp. 1–5.

[38] V. Mathews and G. Sicuranza, *Polynomial Signal Processing* (Wiley, New York, 2000).

[39] A. Fermo, A. Carini, and G. L. Sicuranza, Low complexity nonlinear adaptive filters for acoustic echo cancellation, European Transactions on Telecommunications **14**, pp. 161–169 (2003).

[40] M. Mohri, A. Rostamizadeh, and A. Tawalkar, *Foundations of Machine Learning*, 2nd ed. (MIT Press, Cambridge, USA, 2018).

[41] C. C. Aggarwal, *Neural Networks and Deep Learning* (Springer, Switzerland, 2018).

[42] R. A. do Prado, R. M. Guedes, F. R. Henriques, F. M. da Costa, L. D. T. J. Tarrataca, and D. B. Haddad, On the analysis of the incremental ℓ_0-LMS algorithm for distributed systems, Circuits, Systems, and Signal Processing **40**, pp. 845–871 (2021).

[43] S. Jiang and Y. Gu, Block-sparsity-induced adaptive filter for multi-clustering system identification, IEEE Transactions on Signal Processing **63**, pp. 5318–5330 (2015).

[44] Y. Li, Z. Jiang, Z. Jin, X. Han, and J. Yin, Cluster-sparse proportionate NLMS algorithm with the hybrid norm constraint, IEEE Access **6**, pp. 47794–47803 (2018).

[45] D. L. Duttweiler, Proportionate normalized least-mean-squares adaptation in echo cancelers, IEEE Transactions on Speech and Audio Processing **8**, pp. 508–518 (2000).

[46] J. Benesty and S. L. Gay, An improved PNLMS algorithm. Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2002), Dallas, USA, May 2002, pp. 1881–1884.

[47] H. Deng and M. Doroslovacki, Improving convergence of the PNLMS algorithm for sparse impulse response identification, IEEE Signal Processing Letters **12**, pp. 181–184 (2005).

[48] E. Hänsler and G. Schmidt, *Acoustic Echo and Noise Control: A Practical Approach* (Wiley, Hoboken, 2004).

[49] L. Ligang, M. Fukumoto, and S. Saiki, An improved mu-law proportionate NLMS algorithm. Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2008), 2008 pp. 3797–3800.

[50] S. Werner, J. A. Apolinário Jr., P. S. R. Diniz, and T. I. Laakso, A set-membership approach to normalized proportionate adaptation algorithms. Proceeding of the European Signal Processing Conference (EUSIPCO 2005), Antalya, Turkey, September 2005, pp. 1–4.

[51] M. V. S. Lima and P. S. R. Diniz, Steady-state MSE performance of the set-membership affine projection algorithm, Circuits, Systems, and Signal Processing **32**, pp. 1811–1837 (2013).

[52] P. A. Naylor, J. Cui, and M. Brookes, Adaptive algorithms for sparse echo cancellation, Signal Processing **86**, pp. 1182–1192 (2006).

[53] F. C. de Souza, O. J. Tobias, R. Seara, and D. R. Morgan, A PNLMS algorithm with individual activation factors, IEEE Transactions on Signal Processing **58**, pp. 2036–2047 (2010).

[54] F. C. de Souza, R. Seara, and D. R. Morgan, An enhanced IAF-PNLMS adaptive algorithm for sparse impulse response identification, IEEE Transactions on Signal Processing **60**, pp. 3301–3307 (2012).

[55] A. W. H. Khong and P. A. Naylor, Efficient use of sparse adaptive filters. Proceedings of the Fortieth Asilomar Conference on Signals, Systems and Computers (ACSSC 2006), 2006, pp. 1375–1379.

[56] C. Paleologu, S. Ciochina, and J. Benesty, An efficient proportionate affine projection algorithm for echo cancellation, IEEE Signal Processing Letters **17**, pp. 165–168 (2010).

[57] Z. Zheng, Z. Liu, and Y. Dong, Steady-State and Tracking Analyses of the Improved Proportionate Affine Projection Algorithm, IEEE Transactions on Circuits and Systems II: Express Briefs **65**, pp. 1793–1797 (2018).

[58] R. Arablouei, K. Dogançay and S. Perreau, Proportionate affine projection algorithm with selective projections for sparse system identification. Proceedings of the Asia-Pacific Signal Information Processing Association Annual Summit Conference, 2010, pp. 362–366.

[59] J. V. G. de Souza, D. B. Haddad, F. R. Henriques, and M. R. Petraglia, Novel proportionate adaptive filters with coefficient vector reusing, Circuits, Systems, and Signal Processing **39**, pp. 2473–2488 (2020).

[60] 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).

[61] H. Yazdanpanah, P. S. R. Diniz, and M. V. S. Lima, Improved simple set-membership affine projection algorithm for sparse system modelling: Analysis and implementation, IET Signal Processing **14**, pp. 81–88 (2020).

[62] K. Nose-Filho, A. K. Takahata, R. Lopes, and J. M. T. Romano, Improving sparse multichannel blind deconvolution with correlated seismic data: Foundations and further results, IEEE Signal Processing Magazine **35**, pp. 41–50 (2018).
[63] Y. Kopsinis, K. Slavakis, and S. Theodoridis, Online sparse system identification and signal reconstruction using projections onto weighted l_1 balls, IEEE Transactions on Signal Processing **59**, pp. 936–952 (2011).
[64] K. S. Olinto, D. B. Haddad, and M. R. Petraglia, Transient analysis of ℓ_0-LMS and ℓ_0-NLMS algorithms, Signal Processing **127**, pp. 217–226 (2016).
[65] M. Pereyra et al., A survey of stochastic simulation and optimization methods in signal processing, IEEE Journal of Selected Topics in Signal Processing **10**, pp. 224–241 (2016).
[66] S. Chouvardas, K. Slavakis, Y. Kopsinis, and S. Theodoridis, A sparsity promoting adaptive algorithm for distributed learning, IEEE Transactions on Signal Processing **60**, pp. 5412–5425 (2012).
[67] T. N. Ferreira, M. V. S. Lima, W. A. Martins, and P. S. R. Diniz, Low-complexity proportionate algorithms with sparsity-promoting penalties. Proceedings of the 2016 IEEE International Symposium on Circuits and Systems, Montreal, Canada, May 2016, pp. 253–256.