CHAPTER

THE LEAST-SQUARES FAMILY

6

CHAPTER OUTLINE

Machine Learning. http://dx.doi.org/10.1016/B978-0-12-801522-3.00006-9

6.1 INTRODUCTION

The squared error loss function was at the center of our attention in the previous two chapters. The sum of error-squares cost was introduced in Chapter 3, followed by the mean-square error (MSE) version, treated in Chapter 4. The stochastic gradient descent technique was employed in Chapter 5 to help us bypass the need to perform expectations for obtaining the second order statistics of the data, as required by the MSE formulation.

In this chapter, we return to the original formulation of the sum of error squares, and our goal is to look more closely at the resulting family of algorithms and their properties. A major part of the chapter will be dedicated to the recursive least-squares (RLS) algorithm, which is an online scheme that solves the least-squares (LS) optimization task. The spine of the RLS scheme comprises an efficient update of the inverse (sample) covariance matrix of the input data, whose rationale can also be adopted in the context of different learning methods for developing related online schemes; this is one of the reasons we pay special tribute to the RLS algorithm. The other reason is its popularity in a large number of signal processing/machine learning tasks, due to some attractive properties that this scheme enjoys.

Finally, at the end of the chapter, a more general formulation of the LS task, known as the total-least-squares (TLS), is given and reviewed.

6.2 LEAST-SQUARES LINEAR REGRESSION: A GEOMETRIC PERSPECTIVE

We begin with our familiar linear regression model. Given a set of observations,

$$y_n = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n + \eta_n, \quad n = 1, 2, \ldots, N,\ y_n \in \mathbb{R},\ \boldsymbol{x}_n \in \mathbb{R}^l,\ \boldsymbol{\theta} \in \mathbb{R}^l,$$

where η_n denotes the (unobserved) values of a *zero* mean noise source, the task is to obtain an estimate of the unknown parameter vector, $\boldsymbol{\theta}$, so that

$$\hat{\boldsymbol{\theta}}_{\mathrm{LS}} = \arg\min_{\boldsymbol{\theta}} \sum_{n=1}^{N} (y_n - \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{x}_n)^2. \tag{6.1}$$

Our stage of discussion is that of real numbers, and we will point out differences with the complex number case whenever needed. Moreover, we assume that our data have been centered around their sample means; alternatively, the intercept, θ_0, can be absorbed in $\boldsymbol{\theta}$ with a corresponding increase in the dimensionality of $\boldsymbol{x}_n$. Define,

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^N, \quad X := \begin{bmatrix} \boldsymbol{x}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{x}_N^{\mathrm{T}} \end{bmatrix} \in \mathbb{R}^{N \times l}. \tag{6.2}$$

Equation (6.1) can be recast as,

$$\hat{\boldsymbol{\theta}}_{\mathrm{LS}} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{e}\|^2,$$

where

$$\boldsymbol{e} := \boldsymbol{y} - X\boldsymbol{\theta},$$

and $\|\cdot\|$ denotes the Euclidean norm, which measures the "distance" between the respective vectors in $\mathbb{R}^N$, i.e., $\boldsymbol{y}$ and $X\boldsymbol{\theta}$. Let us denote as $\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_l^c \in \mathbb{R}^N$ the columns of X, i.e.,

$$X = [\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_l^c].$$

Then we can write,

$$\hat{\boldsymbol{y}} := X\boldsymbol{\theta} = \sum_{i=1}^{l} \theta_i \boldsymbol{x}_i^c,$$

and

$$\boldsymbol{e} = \boldsymbol{y} - \hat{\boldsymbol{y}}.$$

Obviously, $\hat{\boldsymbol{y}}$ represents a vector that lies in the span$\{\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_l^c\}$. Thus, naturally, our task now becomes that of selecting $\boldsymbol{\theta}$ so that the error vector between $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ has minimum norm. According to the Pythagorean theorem of orthogonality for Euclidean spaces this is achieved if $\hat{\boldsymbol{y}}$ is chosen as the orthogonal projection of $\boldsymbol{y}$ onto the span$\{\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_l^c\}$. Figure 6.1 illustrates the geometry. Recalling the concept of orthogonal projections (Section 5.6, Eq. (5.64)), we obtain

$$\boxed{\hat{\boldsymbol{y}} = X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}\boldsymbol{y}: \quad \text{LS Estimate,}} \tag{6.3}$$

assuming that $X^{\mathrm{T}}X$ is invertible. It is common to describe the LS solution in terms of the *Moore-Penrose pseudo-inverse* of X, which for a tall matrix is defined as,

$$\boxed{X^{\dagger} := (X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}: \quad \text{Pseudo-inverse of a Tall Matrix } X,} \tag{6.4}$$

and hence we can write,

$$\hat{\boldsymbol{\theta}}_{\mathrm{LS}} = X^{\dagger}\boldsymbol{y}. \tag{6.5}$$

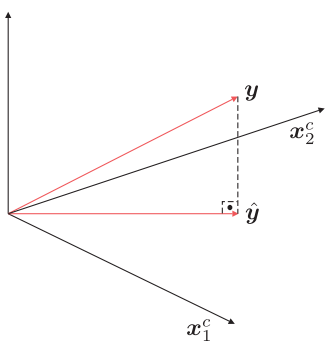

FIGURE 6.1

The LS estimate is chosen so that $\hat{\boldsymbol{y}}$ is the orthogonal projection of $\boldsymbol{y}$ onto the span$\{\boldsymbol{x}_1^c, \boldsymbol{x}_2^c\}$, that is the columns of X.

Thus, we have rederived Eq. (3.17) of Chapter 3, this time via geometric arguments. Note that the pseudo-inverse is a generalization of the notion of the inverse of a square matrix. Indeed, if X is square, then it is readily seen that the pseudo inverse coincides with X^{-1}. For complex-valued data, the only difference is that transposition is replaced by the Hermitian one.

6.3 STATISTICAL PROPERTIES OF THE LS ESTIMATOR

Some of the statistical properties of the LS estimator were touched on in Chapter 3, for the special case of a random real parameter. Here, we will look at this issue in a more general setting. Assume that there exists a true (yet unknown) parameter/weight vector, $\boldsymbol{\theta}_o$, that generates the output (dependent) random variables (stacked in a random vector $\mathbf{y} \in \mathbb{R}^N$), according to the model,

$$\mathbf{y} = X\boldsymbol{\theta}_o + \boldsymbol{\eta},$$

where $\boldsymbol{\eta}$ is a *zero* mean noise vector. Observe that we have assumed that X is fixed and not random; that is, the randomness underlying the output variables, $\mathbf{y}$, is due solely to the noise. Under the previously stated assumptions, the following properties hold.

The LS estimator is unbiased

The LS estimator for the parameters is given by

$$\begin{aligned}\hat{\theta}_{\text{LS}} &= (X^{\text{T}}X)^{-1}X^{\text{T}}\mathbf{y},\\ &= (X^{\text{T}}X)^{-1}X^{\text{T}}(X\boldsymbol{\theta}_o + \boldsymbol{\eta}) = \boldsymbol{\theta}_o + (X^{\text{T}}X)^{-1}X^{\text{T}}\boldsymbol{\eta},\end{aligned} \tag{6.6}$$

or

$$\mathbb{E}[\hat{\theta}_{\text{LS}}] = \boldsymbol{\theta}_o + (X^{\text{T}}X)^{-1}X^{\text{T}}\,\mathbb{E}[\boldsymbol{\eta}] = \boldsymbol{\theta}_o,$$

which proves the claim.

Covariance matrix of the LS estimator

Let in addition to the previously adopted assumptions that

$$\mathbb{E}[\boldsymbol{\eta}\boldsymbol{\eta}^{\text{T}}] = \sigma_\eta^2 I,$$

that is, the source generating the noise samples is white. By the definition of the covariance matrix, we get

$$\Sigma_{\hat{\theta}_{\text{LS}}} = \mathbb{E}\left[(\hat{\theta}_{\text{LS}} - \boldsymbol{\theta}_o)(\hat{\theta}_{\text{LS}} - \boldsymbol{\theta}_o)^{\text{T}}\right],$$

and substituting $\hat{\theta}_{\text{LS}} - \boldsymbol{\theta}_o$ from (6.6), we obtain

$$\begin{aligned}\Sigma_{\hat{\theta}_{\text{LS}}} &= \mathbb{E}\left[(X^{\text{T}}X)^{-1}X^{\text{T}}\boldsymbol{\eta}\boldsymbol{\eta}^{\text{T}}X(X^{\text{T}}X)^{-1}\right]\\ &= (X^{\text{T}}X)^{-1}X^{\text{T}}\,\mathbb{E}[\boldsymbol{\eta}\boldsymbol{\eta}^{\text{T}}]X(X^{\text{T}}X)^{-1}\\ &= \sigma_\eta^2(X^{\text{T}}X)^{-1}.\end{aligned} \tag{6.7}$$

Note that, for large values of N, we can write

$$X^TX = \sum_{n=1}^{N} \boldsymbol{x}_n\boldsymbol{x}_n^T \approx N\Sigma_x,$$

where Σ_x is the covariance matrix of our (zero mean) input variables, i.e.,

$$\Sigma_x := \mathbb{E}[\mathbf{x}_n \mathbf{x}_n^T] \approx \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^T.$$

Thus, for large values of N, we can write

$$\boxed{\Sigma_{\hat{\theta}_{LS}} \approx \frac{\sigma_\eta^2}{N} \Sigma_x^{-1}.} \tag{6.8}$$

In other words, under the adopted assumptions, the LS estimator is not only unbiased, but its covariance matrix *tends asymptotically to zero*. That is, with high probability, the estimate $\hat{\boldsymbol{\theta}}_{\mathrm{LS}}$, which is obtained via a large number of measurements, will be close to the true value $\boldsymbol{\theta}_o$. Viewing it slightly differently, note that the LS solution tends to the MSE solution, which was discussed in Chapter 4. Indeed, for the case of centered data,

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^T = \Sigma_x,$$

and

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n y_n = \mathbb{E}[\mathbf{x}\mathrm{y}] = \boldsymbol{p}.$$

Moreover, we know that for the linear regression modeling case, the normal equations, $\Sigma_x \boldsymbol{\theta} = \boldsymbol{p}$. result in the solution $\boldsymbol{\theta} = \boldsymbol{\theta}_o$ (Remarks 4.2).

The LS estimator is BLUE in the presence of white noise

The notion of the best linear unbiased estimator (BLUE) was introduced in Section 4.9.1 in the context of the *Gauss Markov* theorem. Let $\hat{\theta}$ denote any other *linear* unbiased estimator, under the assumption that

$$\mathbb{E}[\boldsymbol{\eta}\boldsymbol{\eta}^{\mathrm{T}}] = \sigma_\eta^2 I.$$

Then, by its definition, the estimator will have a linear dependence on the output data, i.e.,

$$\hat{\theta} = H\mathbf{y}, \quad H \in \mathbb{R}^{l\times N}.$$

It will be shown that

$$\mathbb{E}\left[(\hat{\theta} - \boldsymbol{\theta}_o)^T (\hat{\theta} - \boldsymbol{\theta}_o)\right] \geq \mathbb{E}\left[(\hat{\theta}_{LS} - \boldsymbol{\theta}_o)^T (\hat{\theta}_{LS} - \boldsymbol{\theta}_o)\right]. \tag{6.8a}$$

Indeed, from the respective definitions we have

$$\hat{\theta} = H(X\boldsymbol{\theta}_o + \boldsymbol{\eta}) = HX\boldsymbol{\theta}_o + H\boldsymbol{\eta}. \tag{6.9}$$

However, because $\hat{\theta}$ has been assumed unbiased, then (6.9) implies that, $HX = I$ and

$$\hat{\theta} - \boldsymbol{\theta}_o = H\boldsymbol{\eta}.$$

Thus,

$$\Sigma_{\hat{\theta}} := \mathbb{E}\left[(\hat{\theta} - \boldsymbol{\theta}_o)(\hat{\theta} - \boldsymbol{\theta}_o)^T\right]$$
$$= \sigma_\eta^2 HH^T.$$

However, taking into account that $HX = I$, it is easily checked out (try it) that

$$\sigma_\eta^2 HH^T = \sigma_\eta^2 (H - X^\dagger)(H - X^\dagger)^T + \sigma_\eta^2 (X^T X)^{-1},$$

where $X^\dagger$ is the respective pseudo-inverse matrix, defined in (6.4).

Because $\sigma_n^2 (H - X^\dagger)(H - X^\dagger)^{\mathrm{T}}$ is a positive semidefinite matrix, its trace is nonnegative (Problem 6.1) and thus we have that

$$\text{trace}\{\sigma_n^2 HH^T\} \geq \text{trace}\{\sigma_n^2 (X^T X)^{-1}\},$$

and recalling (6.7), Eq. (6.8a) is proved, with equality only if

$$H = X^\dagger = (X^{\mathrm{T}} X)^{-1} X^{\mathrm{T}}.$$

Note that this result could have been obtained directly from (4.102) by setting $\Sigma_\eta = \sigma_\eta^2 I$. This also emphasizes the fact that if the noise is not white, then the LS parameter estimator is *no more* BLUE.

The LS estimator achieves the Cramér-Rao bound for white Gaussian noise

The concept of the Cramér-Rao lower bound was introduced in Chapter 3. There, it was shown that, under the *white Gaussian noise* assumption, the LS estimator of a real number was *efficient*; that is, it achieves the CR bound. Moreover, in Problem 3.8, it was shown that if $\boldsymbol{\eta}$ is zero mean Gaussian noise with covariance matrix Σ_η, then the efficient estimator is given by

$$\hat{\theta} = (X^T \Sigma_\eta^{-1} X)^{-1} X^T \Sigma_\eta^{-1} \mathbf{y},$$

which for $\Sigma_\eta = \sigma_\eta^2 I$ coincides with the LS estimator. In other words, under the white Gaussian noise assumption, the LS estimator becomes *minimum variance unbiased estimator* (MVUE). This is a strong result. No other unbiased estimator (*not necessarily linear*) will do better than the LS one. Note that this result holds true not asymptotically, but also for finite number of samples N. If one wishes to decrease further the mean-square error, then *biased* estimators, as produced via regularization, have to be considered; this has already been discussed in Chapter 3; see also [16, 50] and the references therein.

Asymptotic distribution of the LS estimator

We have already seen that the LS estimator is unbiased and that its covariance matrix is approximately (for large values of N) given by (6.8). Thus, as $N \longrightarrow \infty$, the variance around the true value, $\boldsymbol{\theta}_o$, is becoming increasingly small. Furthermore, there is a stronger result, which provides the distribution of the LS estimator for large values of N. Under some general assumptions, such as independence of successive observation vectors and that the white noise source is independent of the input, and mobilizing the central limit theorem, it can be shown (Problem 6.2) that

$$\boxed{\sqrt{N}(\hat{\theta}_{LS} - \boldsymbol{\theta}_o) \longrightarrow \mathcal{N}(\mathbf{0}, \sigma^2 \Sigma_x^{-1}),} \tag{6.10}$$

where the limit is meant to be in *distribution* (see Section 2.6). Alternatively, we can write that

$$\hat{\theta}_{LS} \sim \mathcal{N}\left(\boldsymbol{\theta}_o, \frac{\sigma^2}{N}\Sigma_x^{-1}\right).$$

In other words, the LS parameter estimator is asymptotically distributed according to the normal distribution.

6.4 ORTHOGONALIZING THE COLUMN SPACE OF X: THE SVD METHOD

The *singular value decomposition* (SVD) of a matrix is among the most powerful tools in linear algebra. Due to its importance in machine learning, we present the basic theory here and exploit it to shed light onto our LS estimation task from a different angle. We start by considering the general case, and then we tailor the theory to our specific needs.

Let X be an $m \times l$ matrix and allow its rank, r, not to be necessarily full, i.e.,

$$r \leq \min(m, l).$$

Then, there exist *orthogonal* matrices,[1] U and V, of dimensions $m \times m$ and $l \times l$, respectively, so that

$$\boxed{X = U\begin{bmatrix} D & O \\ O & O \end{bmatrix}V^T: \quad \text{Singular Value Decomposition of } X,} \tag{6.11}$$

where D is an $r \times r$ *diagonal* matrix[2] with elements $\sigma_i = \sqrt{\lambda_i}$, known as the *singular values* of X, where $\lambda_i,\ i = 1, 2, \ldots, r$, are the *nonzero* eigenvalues of XX^{T}; matrices denoted as O comprise zero elements and are of appropriate dimensions.

Taking into account the zero elements in the diagonal matrix, (6.11) can be rewritten as

$$X = U_r D V_r^{\mathrm{T}} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}, \tag{6.12}$$

where

$$U_r := [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r] \in \mathbb{R}^{m \times r}, \quad V_r := [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r] \in \mathbb{R}^{l \times r}. \tag{6.13}$$

Equation (6.12) provides a matrix factorization of X in terms of U_r, V_r, and D. We will make use of this factorization in Chapter 19, when dealing with dimensionality reduction techniques. Figure 6.2 offers a schematic illustration of (6.12).

It turns out that $\boldsymbol{u}_i \in \mathbb{R}^m,\ i = 1, 2, \ldots, r$, known as *left singular vectors*, are the normalized eigenvectors corresponding to the nonzero eigenvalues of XX^{T}, and $\boldsymbol{v}_i \in \mathbb{R}^l,\ i = 1, 2, \ldots, r$, are the normalized eigenvectors associated with the nonzero eigenvalues of $X^{\mathrm{T}}X$, and they are known as *right singular vectors*. Note that both XX^{T} and $X^{\mathrm{T}}X$ share the same eigenvalues (Problem 6.3).

[1] Recall that a square matrix U is called orthogonal if $U^T U = UU^T = I$. For complex-valued square matrices, if $U^H U = UU^H = I$, it is called unitary.

[2] Usually it is denoted as Σ, but here we avoid the notation so as not to confuse it with the covariance matrix Σ; D reminds us of its diagonal structure.

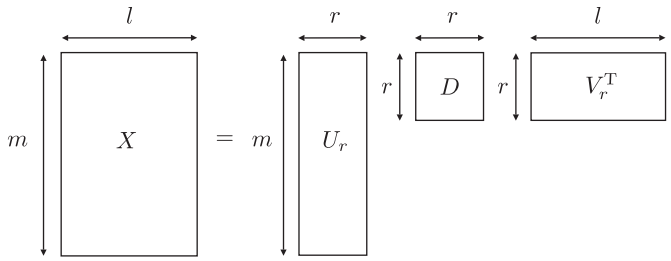

FIGURE 6.2

The $m \times l$ matrix X, of rank $r \leq \min(m, l)$, factorizes in terms of the matrices $U_r \in \mathbb{R}^{m \times r}$, $V_r \in \mathbb{R}^{l \times r}$ and the $r \times r$ diagonal matrix D.

Proof. By the respective definitions, we have

$$XX^{\mathrm{T}}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i, \quad i = 1, 2, \ldots, r, \tag{6.14}$$

and

$$X^{\mathrm{T}}X\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i, \quad i = 1, 2, \ldots, r. \tag{6.15}$$

Moreover, because XX^{T} and $X^{\mathrm{T}}X$ are symmetric matrices, it is known from linear algebra that their eigenvalues are real[3] and the respective eigenvectors are orthogonal, which can then be normalized to unit norm to become orthonormal (Problem 6.4). It is a matter of simple algebra (Problem 6.5) to show from (6.14) and (6.15) that,

$$\boldsymbol{u}_i = \frac{1}{\sigma_i} X\boldsymbol{v}_i, \quad i = 1, 2, \ldots, r. \tag{6.16}$$

Thus, we can write that

$$\sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}} = X \sum_{i=1}^{r} \boldsymbol{v}_i \boldsymbol{v}_i^{\mathrm{T}} = X \sum_{i=1}^{l} \boldsymbol{v}_i \boldsymbol{v}_i^{\mathrm{T}} = XVV^{\mathrm{T}},$$

where we used the fact that for eigenvectors corresponding to $\sigma_i = 0$ ($\lambda_i = 0$), $i = r+1, \ldots, l$, $X\boldsymbol{v}_i = \boldsymbol{0}$. However, due to the orthonormality of $\boldsymbol{v}_i$, $i = 1, 2, \ldots, l$, $VV^{\mathrm{T}} = I$ and the claim in (6.12) has been proved. □

Pseudo-inverse matrix and SVD

Let us now elaborate on the SVD expansion. By the definition of the pseudo-inverse, $X^{\dagger}$, and assuming the $N \times l$ ($N > l$) data matrix to be full column rank ($r = l$), then employing (6.12) in (6.5) we get (Problem 6.6),

$$\begin{aligned} \hat{\boldsymbol{y}} &= X\hat{\boldsymbol{\theta}}_{\mathrm{LS}} = X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}\boldsymbol{y} \\ &= U_l U_l^{\mathrm{T}}\boldsymbol{y} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_l] \begin{bmatrix} \boldsymbol{u}_1^{\mathrm{T}}\boldsymbol{y} \\ \vdots \\ \boldsymbol{u}_l^{\mathrm{T}}\boldsymbol{y} \end{bmatrix}, \end{aligned}$$

[3] This is also true for complex matrices, XX^{H}, $X^{\mathrm{H}}X$.

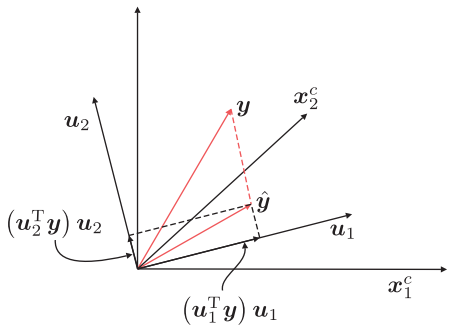

FIGURE 6.3

The eigenvectors $\boldsymbol{u}_1, \boldsymbol{u}_2$, from an orthonormal basis, in span$\{\boldsymbol{x}_1^c, \boldsymbol{x}_2^c\}$; that is, the column space of X. $\hat{\boldsymbol{y}}$ is the projection of $\boldsymbol{y}$ onto this subspace.

or

$$\boxed{\hat{\boldsymbol{y}} = \sum_{i=1}^{l} (\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{y})\boldsymbol{u}_i : \quad \text{LS Estimate in Terms of an Orthonormal Basis.}} \tag{6.17}$$

The latter represents the *projection* of $\boldsymbol{y}$ onto the column space of X, i.e., span$\{\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_l^c\}$ using a corresponding *orthonormal* basis, $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_l\}$, to describe the subspace, see Figure 6.3. Note that each $\boldsymbol{u}_i$, $i = 1, 2, \ldots, l$, lies in the space spanned by the columns of X as it is suggested from Eq. (6.16). Moreover, it is easily shown that we can write

$$X^{\dagger} = (X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}} = V_l D^{-1} U_l^{\mathrm{T}} = \sum_{i=1}^{l} \frac{1}{\sigma_i} \boldsymbol{v}_i \boldsymbol{u}_i^{\mathrm{T}}.$$

As a matter of fact, this is in line with the more general definition of a pseudo-inverse in linear algebra, including matrices that are not full rank (i.e., $X^{\mathrm{T}}X$ is not invertible), namely

$$\boxed{X^{\dagger} := V_r D^{-1} U_r^{\mathrm{T}} = \sum_{i=1}^{r} \frac{1}{\sigma_i} \boldsymbol{v}_i \boldsymbol{u}_i^{\mathrm{T}} : \quad \text{Pseudo-inverse of a Matrix of Rank } r.} \tag{6.18}$$

In the case of matrices with $N < l$, and assuming that the rank of X is equal to N, then it is readily verified that the previous generalized definition of the pseudo-inverse is equivalent to

$$\boxed{X^{\dagger} = X^{\mathrm{T}}(XX^{\mathrm{T}})^{-1} : \quad \text{Pseudo-inverse of a Fat Matrix, } X.} \tag{6.19}$$

Note that a system with N equations and $l > N$ unknowns,

$$X\boldsymbol{\theta} = \boldsymbol{y},$$

has infinite solutions. Such systems are known as *underdetermined*, to be contrasted with the *over-determined* systems for which $N > l$. It can be shown that for underdetermined systems, the solution $\boldsymbol{\theta} = X^{\dagger}\boldsymbol{y}$ is the one with the minimum Euclidean norm. We will consider the case of such systems of equations in more detail in Chapter 9, in the context of sparse models.

Remarks 6.1.

- Computing the pseudo-inverse using the SVD is numerically more robust than the direct method via the inversion of $(X^{\mathrm{T}}X)^{-1}$.
- *k-rank matrix approximation*: The best rank $k < r \leq \min(m, l)$ approximation matrix, $\hat{X} \in \mathbb{R}^{m\times l}$, of $X \in \mathbb{R}^{m\times l}$ in the Frobenius, $\|\cdot\|_F$, as well as in the spectral, $\|\cdot\|_2$, norms sense is given by (e.g., [26]),

$$\hat{X} = \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}, \tag{6.20}$$

with the previously stated norms defined as (Problem 6.9),

$$\boxed{\|X\|_F := \sqrt{\sum_i \sum_j |X(i,j)|^2} = \sqrt{\sum_{i=1}^{r} \sigma_i^2} : \quad \text{Frobenius Norm of } X,} \tag{6.21}$$

and

$$\boxed{\|X\|_2 := \sigma_1 : \quad \text{Spectral Norm of } X,} \tag{6.22}$$

where, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$ are the singular values of X. In other words, $\hat{X}$ in (6.20) minimizes the error matrix norms,

$$\|X - \hat{X}\|_F \quad \text{and} \quad \|X - \hat{X}\|_2.$$

Moreover, it turns out that the approximation error is given by (Problems 6.10 and 6.11),

$$\|X - \hat{X}\|_F = \sqrt{\sum_{i=k+1}^{r} \sigma_i^2}, \quad \|X - \hat{X}\|_2 = \sigma_{k+1}.$$

This is also known as the *Eckart-Young-Mirsky* theorem.

- *Null and range spaces of X*: Let the rank of an $m \times l$ matrix, X, be equal to $r \leq \min(m, l)$. Then, the following easily shown properties hold (Problem 6.13): The null space of X, $\mathcal{N}(X)$, defined as

$$\mathcal{N}(X) := \{\boldsymbol{x} \in \mathbb{R}^l : X\boldsymbol{x} = \boldsymbol{0}\}, \tag{6.23}$$

is also expressed as

$$\mathcal{N}(X) = \text{span}\{\boldsymbol{v}_{r+1}, \ldots, \boldsymbol{v}_l\}. \tag{6.24}$$

Furthermore, the range space of X, $\mathcal{R}(X)$, defined as

$$\mathcal{R}(X) := \{\boldsymbol{x} \in \mathbb{R}^l : \exists\, \boldsymbol{a} \text{ such as } X\boldsymbol{a} = \boldsymbol{x}\}, \tag{6.25}$$

is expressed as

$$\mathcal{R}(X) = \text{span}\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r\}. \tag{6.26}$$

- Everything that has been said before transfers to complex-valued data, trivially, by replacing transposition with the Hermitian one.

6.5 RIDGE REGRESSION

Ridge regression was introduced in Chapter 3 as a means to impose bias on the LS solution and also as a major path to cope with overfitting and ill-conditioning problems. In ridge regression, the minimizer results as

$$\hat{\boldsymbol{\theta}}_R = \arg\min_{\boldsymbol{\theta}} \left\{ \|\boldsymbol{y} - X\boldsymbol{\theta}\|^2 + \lambda \|\boldsymbol{\theta}\|^2 \right\},$$

where $\lambda > 0$ is a user-defined parameter that controls the importance of the regularizing term. Taking the gradient w.r.t. $\boldsymbol{\theta}$ and equating to zero results in

$$\hat{\boldsymbol{\theta}}_R = (X^{\mathrm{T}}X + \lambda I)^{-1} X^{\mathrm{T}}\boldsymbol{y}. \tag{6.27}$$

Looking at (6.27), we readily observe (a) its "stabilizing" effect from the numerical point of view, when $X^{\mathrm{T}}X$ has large condition number, and (b) its biasing effect on the (unbiased) LS solution. Note that ridge regression provides a solution even if $X^{\mathrm{T}}X$ is not invertible, as is the case when $N < l$. Let us now employ the SVD expansion of (6.12) in (6.27). Assuming a full column rank matrix, X, we obtain (Problem 6.14),

$$\hat{\boldsymbol{y}} = X\hat{\boldsymbol{\theta}}_R = U_l D(D^2 + \lambda I)^{-1} D U_l^{\mathrm{T}} \boldsymbol{y},$$

or

$$\boxed{\hat{\boldsymbol{y}} = \sum_{i=1}^{l} \frac{\sigma_i^2}{\lambda + \sigma_i^2} (\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{y})\boldsymbol{u}_i : \quad \text{Ridge Regression Shrinks the Weights.}} \tag{6.28}$$

Comparing (6.28) and (6.17), we observe that the components of the projection of $\boldsymbol{y}$ onto the $\mathrm{span}\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_l\}$ ($\mathrm{span}\{\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_l^c\}$) are *shrunk* with respect to their LS counterpart. Moreover, the shrinking level depends on the singular values, σ_i; the smaller the value of σ_i, the higher the shrinking of the corresponding component. Let us now turn our attention to investigate the geometric interpretation of this algebraic finding. This small diversion will also provide more insight in the interpretation of $\boldsymbol{v}_i$ and $\boldsymbol{u}_i$, $i = 1, 2, \ldots, l$, that appear in the SVD method.

Recall that $X^{\mathrm{T}}X$ is a scaled version of the sample covariance matrix for centered regressors. Also, by the definition of the $\boldsymbol{v}_i$'s, we have

$$(X^{\mathrm{T}}X)\boldsymbol{v}_i = \sigma_i^2 \boldsymbol{v}_i, \quad i = 1, 2, \ldots, l,$$

and in a compact form,

$$(X^{\mathrm{T}}X)V_l = V_l \,\mathrm{diag}\{\sigma_1^2, \ldots, \sigma_l^2\} \Rightarrow$$

$$(X^{\mathrm{T}}X) = V_l D^2 V_l^{\mathrm{T}} = \sum_{i=1}^{l} \sigma_i^2 \boldsymbol{v}_i \boldsymbol{v}_i^{\mathrm{T}}. \tag{6.29}$$

Note that in (6.29), the (scaled) sample covariance matrix is written as a sum of rank one matrices, $\boldsymbol{v}_i \boldsymbol{v}_i^{\mathrm{T}}$, each one weighted by the square of respective singular value, σ_i^2. We are now close to revealing the physical/geometric meaning of the singular values. To this end, define

$$\boldsymbol{q}_j := X\boldsymbol{v}_j = \begin{bmatrix} \boldsymbol{x}_1^{\mathrm{T}}\boldsymbol{v}_j \\ \vdots \\ \boldsymbol{x}_N^{\mathrm{T}}\boldsymbol{v}_j \end{bmatrix} \in \mathbb{R}^N, \quad j = 1, 2, \ldots, l. \tag{6.30}$$

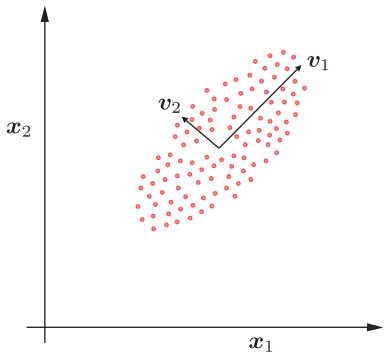

FIGURE 6.4

The singular vector $\boldsymbol{v}_1$, which is associated with the the singular value $\sigma_1 > \sigma_2$, points to the direction where most of the (variance) activity in the data space takes place. The variance in the direction of $\boldsymbol{v}_2$ is smaller.

Note that $\boldsymbol{q}_j$ is a vector in the column space of X. Moreover, the respective squared norm of $\boldsymbol{q}_j$ is given by

$$\sum_{n=1}^{N} q_j^2(n) = \boldsymbol{q}_j^{\mathrm{T}}\boldsymbol{q}_j = \boldsymbol{v}_j^{\mathrm{T}}X^{\mathrm{T}}X\boldsymbol{v}_j = \boldsymbol{v}_j^{\mathrm{T}}\left(\sum_{i=1}^{l}\sigma_i^2\boldsymbol{v}_i\boldsymbol{v}_i^{\mathrm{T}}\right)\boldsymbol{v}_j = \sigma_j^2,$$

due to the orthonormality of the $\boldsymbol{v}_j$'s. That is, σ_j^2 is equal to the (scaled) sample variance of the elements of $\boldsymbol{q}_j$. However, by the definition in (6.30), this is the sample variance of the projections of the input vectors (regressors), $\boldsymbol{x}_n$, $n = 1, 2, \ldots, N$, along the direction $\boldsymbol{v}_j$. The larger the value of σ_j, the larger the spread of the (input) data along the respective direction. This is shown in Figure 6.4, where $\sigma_1 \gg \sigma_2$. From the variance point of view, $\boldsymbol{v}_1$ is the more *informative* direction, compared to $\boldsymbol{v}_2$. It is the direction where most of the activity takes place. This observation is at the heart of *dimensionality reduction*, which will be treated in more detail in Chapter 19. Moreover, from (6.16), we obtain

$$\boldsymbol{q}_j = X\boldsymbol{v}_j = \sigma_j\boldsymbol{u}_j. \tag{6.31}$$

In other words, $\boldsymbol{u}_j$ points in the direction of $\boldsymbol{q}_j$. Thus, (6.28) suggests that while projecting $\boldsymbol{y}$ onto the column space of X, the directions, $\boldsymbol{u}_j$, associated with larger values of variance are weighted more heavily than the rest. *Ridge regression respects and assigns higher weights to the more informative directions, where most of the data activity takes place*. Alternatively, the less important directions, those associated with small data variance, are shrunk the most.

One final comment concerning ridge regression is that the ridge solutions are not invariant under scaling of the input variables. This becomes obvious by looking at the respective equations. Thus, in practice, often the input variables are standardized to unit variances.

Principal components regression

We have just seen that the effect of the ridge regression is to enforce a shrinking rule on the parameters, which decreases the contribution of the less important of the components, $\boldsymbol{u}_i$, in the respective summation. This can be considered as a soft shrinkage rule. An alternative path is to adopt a hard

thresholding rule and keep only the m most significant directions, known as the *principal axes* or *directions*, and forget the rest by setting the respective weights equal to zero. Equivalently, we can write

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{m} \hat{\theta}_i \boldsymbol{u}_i, \tag{6.32}$$

where

$$\hat{\theta}_i = \boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{y}, \quad i = 1, 2, \ldots, m. \tag{6.33}$$

Furthermore, employing (6.16) we have that

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{m} \frac{\hat{\theta}_i}{\sigma_i} X \boldsymbol{v}_i, \tag{6.34}$$

or equivalently, the weights for the expansion of the solution in terms of the input data can be expressed as

$$\boldsymbol{\theta} = \sum_{i=1}^{m} \frac{\hat{\theta}_i}{\sigma_i} \boldsymbol{v}_i. \tag{6.35}$$

In other words, the prediction $\hat{\boldsymbol{y}}$ is performed in a subspace of the column space of X, which is spanned by the m principal axes; that is, the subspace where most of the data activity takes place.

6.6 THE RECURSIVE LEAST-SQUARES ALGORITHM

In previous chapters, we discussed the need for developing recursive algorithms that update the estimates every time a new pair of input-output training samples is received. Solving the LS problem using a general purpose solver would amount to $\mathcal{O}(l^3)$ MADS, due to the involved matrix inversion. Also, $\mathcal{O}(Nl^2)$ operations are required to compute the (scaled) sample covariance matrix $X^{\mathrm{T}}X$. In this section, the special structure of $X^{\mathrm{T}}X$ will be taken into account in order to obtain a computationally efficient online scheme for the solution of the LS task. Moreover, when dealing with time recursive techniques, one can also care for time variations of the statistical properties of the involved data. In this section, we will allow for such applications, and the LS cost will be slightly modified in order to accommodate time varying environments.

For the purpose of the section, we will slightly "enrich" our notation and we will use explicitly the time index, n. Also, to be consistent with the previously discussed online schemes, we will assume that the time starts at $n = 0$ and the received observations are $(y_n, \boldsymbol{x}_n)$, $n = 0, 1, 2, \ldots$. To this end, let us denote the input matrix at time n as

$$X_n^{\mathrm{T}} = [\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n].$$

Moreover, the least-squares cost function in (6.1) is modified to involve a *forgetting factor*, $0 < \beta \leq 1$. The purpose of its presence is to help the cost function slowly forget past data samples by weighting heavier the more recent observations. This will equip the algorithm with the ability to track changes that occur in the underlying data statistics. Moreover, since we are interested in time recursive solutions, starting from time $n = 0$, we are forced to introduce regularization. During the

initial period, corresponding to time instants $n < l - 1$, the corresponding system of equations will be underdetermined and $X_n^{\mathrm{T}} X_n$ is not invertible. Indeed, we have that

$$X_n^{\mathrm{T}} X_n = \sum_{i=0}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^{\mathrm{T}}.$$

In other words, $X_n^{\mathrm{T}} X_n$ is the sum of rank one matrices. Hence, for $n < l - 1$ its rank is necessarily less than l, and it cannot be inverted. For larger values of n, it can become full rank, provided that at least l of the input vectors are linearly independent, which is usually assumed to be the case. The previous arguments lead to the following modifications of the "conventional" least-squares, known as the *exponentially weighted least-squares* cost function, minimized by

$$\boxed{\boldsymbol{\theta}_n = \arg\min_{\boldsymbol{\theta}} \left(\sum_{i=0}^{n} \beta^{n-i} (y_i - \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{x}_i)^2 + \lambda \beta^{n+1} \|\boldsymbol{\theta}\|^2 \right),} \tag{6.36}$$

where β is a user-defined parameter, very close to unity, as when $\beta = 0.999$. In this way, the more recent samples are weighted heavier than the older ones. Note that the regularizing parameter has been made time varying. This is because for large values of n, no regularization is required. Indeed for $n > l$, matrix $X_n^{\mathrm{T}} X_n$ becomes, in general, invertible. Moreover, recall from Chapter 3 that the use of regularization also takes precautions for overfitting. However, for very large values of $n \gg l$, this is not a problem, and one wishes to get rid of the imposed bias. The parameter $\lambda > 0$ is also a user-defined variable and its choice will be discussed later on.

Minimizing (6.36) results in

$$\Phi_n \boldsymbol{\theta}_n = \boldsymbol{p}_n, \tag{6.37}$$

where,

$$\Phi_n = \sum_{i=0}^{n} \beta^{n-i} \boldsymbol{x}_i \boldsymbol{x}_i^{\mathrm{T}} + \lambda \beta^{n+1} I, \tag{6.38}$$

and

$$\boldsymbol{p}_n = \sum_{i=0}^{n} \beta^{n-i} \boldsymbol{x}_i y_i, \tag{6.39}$$

which for $\beta = 1$ coincides with the ridge regression.

Time-iterative computations of Φ_n, $\boldsymbol{p}_n$

By the respective definitions, we have that

$$\Phi_n = \beta \Phi_{n-1} + \boldsymbol{x}_n \boldsymbol{x}_n^{\mathrm{T}}, \tag{6.40}$$

and

$$\boldsymbol{p}_n = \beta \boldsymbol{p}_{n-1} + \boldsymbol{x}_n y_n. \tag{6.41}$$

Recall Woodburry's matrix inversion formula (Appendix A.1),

$$(A + BD^{-1}C)^{-1} = A^{-1} - A^{-1}B(D + CA^{-1}B)^{-1}CA^{-1}.$$

Plugging it in (6.40) and after the appropriate inversion and substitutions we obtain,

$$\Phi_n^{-1} = \beta^{-1}\Phi_{n-1}^{-1} - \beta^{-1}K_n\boldsymbol{x}_n^{\mathrm{T}}\Phi_{n-1}^{-1}, \tag{6.42}$$

$$K_n = \frac{\beta^{-1}\Phi_{n-1}^{-1}\boldsymbol{x}_n}{1+\beta^{-1}\boldsymbol{x}_n^{\mathrm{T}}\Phi_{n-1}^{-1}\boldsymbol{x}_n}. \tag{6.43}$$

K_n is known as the *Kalman gain*. For notational convenience, define

$$P_n = \Phi_n^{-1}.$$

Also, rearranging the terms in (6.43), we get

$$K_n = \left(\beta^{-1}P_{n-1} - \beta^{-1}K_n\boldsymbol{x}_n^{\mathrm{T}}P_{n-1}\right)\boldsymbol{x}_n,$$

and taking into account (6.42) results in

$$K_n = P_n\boldsymbol{x}_n. \tag{6.44}$$

Time updating of $\boldsymbol{\theta}_n$

From (6.37), (6.41)–(6.43) we obtain

$$\begin{aligned}\boldsymbol{\theta}_n &= \left(\beta^{-1}P_{n-1} - \beta^{-1}K_n\boldsymbol{x}_n^{\mathrm{T}}P_{n-1}\right)\beta\boldsymbol{p}_{n-1} + P_n\boldsymbol{x}_n y_n\\ &= \boldsymbol{\theta}_{n-1} - K_n\boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{\theta}_{n-1} + K_n y_n,\end{aligned}$$

and finally,

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + K_n e_n, \tag{6.45}$$

where,

$$e_n := y_n - \boldsymbol{\theta}_{n-1}^{\mathrm{T}}\boldsymbol{x}_n. \tag{6.46}$$

The derived algorithm is summarized in Algorithm 6.1.

Algorithm 6.1 .[(The RLS algorithm)]

- Initialize
  - $\boldsymbol{\theta}_{-1} = \mathbf{0}$; any other value is also possible.
  - $P_{-1} = \lambda^{-1}I$; $\lambda > 0$ a user-defined variable.
  - Select β; close to 1.
- **For** $n = 0, 1, \ldots,$ **Do**
  - $e_n = y_n - \boldsymbol{\theta}_{n-1}^{\mathrm{T}}\boldsymbol{x}_n$
  - $\boldsymbol{z}_n = P_{n-1}\boldsymbol{x}_n$
  - $K_n = \frac{\boldsymbol{z}_n}{\beta + \boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{z}_n}$
  - $\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + K_n e_n$
  - $P_n = \beta^{-1}P_{n-1} - \beta^{-1}K_n\boldsymbol{z}_n^{\mathrm{T}}$
- **End For**

Remarks 6.2.

- The complexity of the RLS algorithm is of the order $\mathcal{O}(l^2)$ per iteration, due to the matrix-product operations. That is, there is an order of magnitude difference compared to the LMS and the other schemes that were discussed in Chapter 5. In other words, RLS does not scale well with dimensionality.
- The RLS algorithm shares similar numerical behavior with the Kalman filter, which was discussed in Section 4.10. P_n may lose its positive definite and symmetric nature, which then leads the algorithm to divergence. To remedy such a tendency, symmetry-preserving versions of the RLS algorithm have been derived; see [65, 68]. Note that the use of $\beta < 1$, has a beneficial effect on the error propagation [30, 34]. In Ref. [58], it is shown that for $\beta = 1$ the error propagation mechanism is of a random walk type, hence the algorithm is unstable. In Ref. [5], it is pointed out that due to numerical errors the term $\frac{1}{\beta + \boldsymbol{x}_n^{\mathrm{T}} P_{n-1} \boldsymbol{x}_n}$ may become negative, leading to divergence. The numerical performance of the RLS becomes a more serious concern in implementations using limited precision, such as fixed point arithmetic. Compared to the LMS, RLS would require the use of higher precision implementations; otherwise, divergence may occur after a few iteration steps. This adds further to its computational disadvantage compared to the LMS.
- The choice of λ in the initialization step has been considered in Ref. [46]. The related theoretical analysis suggests that λ has a direct influence on the convergence speed, and it should be chosen so as to be a small positive for high Signal-to-Noise (SNR) ratios and a large positive constant for low SNRs.
- In Ref. [56], it has been shown that the RLS algorithm can be obtained as a special case of the Kalman filter.
- The main advantage of the RLS is that it converges to the steady state much faster than the LMS and the rest of the members of the LMS family. This can be justified by the fact that the RLS can been seen as an offspring of Newton's iterative optimization method.
- Distributed versions of the RLS have been proposed in Refs. [8, 39, 40].

6.7 NEWTON'S ITERATIVE MINIMIZATION METHOD

The steepest descent formulation was presented in Chapter 5. It was noted that it exhibits a linear convergence rate and a heavy dependence on the condition number of the Hessian matrix associated with the cost function. Newton's method is a way to overcome this dependence on the condition number and at the same time improve upon the rate of convergence toward the solution.

In Section 5.2, a first order Taylor expansion was used around the current value $J\left(\boldsymbol{\theta}^{(i-1)}\right)$. Let us now consider a second order expansion (assume $\mu_i = 1$),

$$J\left(\boldsymbol{\theta}^{(i-1)} + \Delta\boldsymbol{\theta}^{(i)}\right) = J\left(\boldsymbol{\theta}^{(i-1)}\right) + \left(\nabla J\left(\boldsymbol{\theta}^{(i-1)}\right)\right)^{\mathrm{T}} \Delta\boldsymbol{\theta}^{(i)}$$

$$+\frac{1}{2}\left(\Delta\boldsymbol{\theta}^{(i)}\right)^{\mathrm{T}} \nabla^2 J\left(\boldsymbol{\theta}^{(i-1)}\right) \Delta\boldsymbol{\theta}^{(i)}.$$

Assuming $\nabla^2 J\left(\boldsymbol{\theta}^{(i-1)}\right)$ to be positive definite (this is always the case if $J(\boldsymbol{\theta})$ is a strictly convex function[4]), the above is a convex quadratic function w.r.t. the step $\Delta\boldsymbol{\theta}^{(i)}$; the latter is computed so as to *minimize* the above second order approximation. The minimum results by equating the corresponding gradient to $\mathbf{0}$, which results in

$$\Delta\boldsymbol{\theta}^{(i)} = -\left(\nabla^2 J\left(\boldsymbol{\theta}^{(i-1)}\right)\right)^{-1} \nabla J\left(\boldsymbol{\theta}^{(i-1)}\right). \tag{6.47}$$

Note that this is indeed a descent direction, because

$$\nabla^T J\left(\boldsymbol{\theta}^{(i-1)}\right) \Delta\boldsymbol{\theta}^{(i)} = -\nabla^T J\left(\boldsymbol{\theta}^{(i-1)}\right) \left(\nabla^2 J\left(\boldsymbol{\theta}^{(i-1)}\right)\right)^{-1} \nabla J\left(\boldsymbol{\theta}^{(i-1)}\right) < 0,$$

due to the positive definite nature of the Hessian; equality to zero is achieved only at a minimum. Thus, the iterative scheme takes the following form:

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} - \mu_i \left(\nabla^2 J\left(\boldsymbol{\theta}^{(i-1)}\right)\right)^{-1} \nabla J\left(\boldsymbol{\theta}^{(i-1)}\right): \quad \text{Newton's Iterative Scheme.} \tag{6.48}$$

Figure 6.5 illustrates the method.

Note that if the cost function is quadratic, then the minimum is achieved at the first iteration!

Observe that in the case of Newton's algorithm, the correction direction is not that of 180° with respect to $\nabla J(\boldsymbol{\theta}^{(i-1)})$, as it is the case for the steepest descent method. An alternative point of view is to look at (6.48) as the steepest descent direction under the following norm (see Section 5.2)

$$\|\boldsymbol{v}\|_P = (\boldsymbol{v}^{\mathrm{T}} P \boldsymbol{v})^{1/2},$$

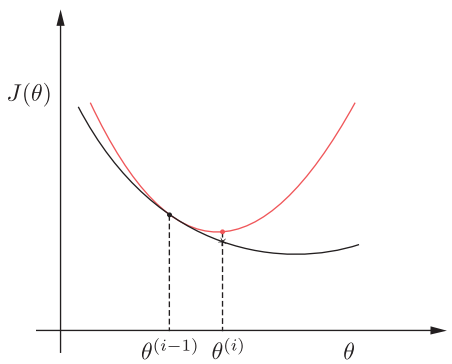

FIGURE 6.5

According to Newton's method, a local quadratic approximation of the cost function is considered (red curve), and the correction pushes the new estimate toward the minimum of this approximation. If the cost function is quadratic, then convergence can be achieved in one step.

[4] See Chapter 8 for related definitions.

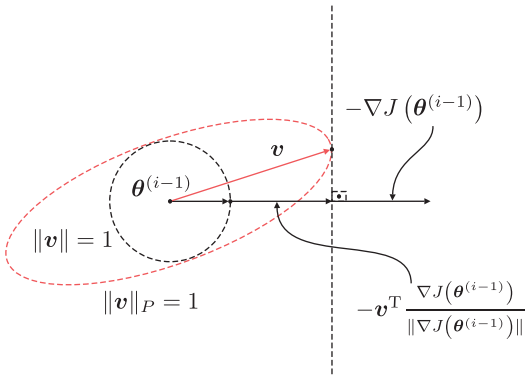

FIGURE 6.6

The graphs of the unit Euclidean (black circle) and quadratic (red ellipse) norms centered at $\boldsymbol{\theta}^{(i-1)}$ are shown. In both cases, the goal is to move as far as possible in the direction of $-\nabla J(\boldsymbol{\theta}^{(i-1)})$, while remaining at the ellipse (circle). The result is different for the two cases. The Euclidean norm corresponds to the steepest descent and the quadratic norm to Newton's method.

where P is a symmetric positive definite matrix. For our case, we set

$$P = \nabla^2 J\left(\boldsymbol{\theta}^{(i-1)}\right).$$

Then, searching for the respective normalized steepest descent direction, i.e.,

$$\begin{aligned} \boldsymbol{v} &= \arg\min_{z} z^T \nabla J\left(\boldsymbol{\theta}^{(i-1)}\right) \\ \text{s.t.} \quad & \|z\|_P^2 = 1, \end{aligned}$$

results to the normalized vector pointing in the same direction as the one in (6.47) (Problem 6.15). For $P = I$, the gradient descent algorithm results. The geometry is illustrated in Figure 6.6. Note that Newton's direction accounts for the *local shape* of the cost function.

The convergence rate for Newton's method is, in general, high and it becomes quadratic close to the solution. Assuming $\boldsymbol{\theta}_*$ to be the minimum, quadratic convergence means that at each iteration, i, the deviation from the optimum value follows the pattern:

$$\boxed{\ln\ln \frac{1}{||\boldsymbol{\theta}^{(i)} - \boldsymbol{\theta}_*||^2} \propto i: \quad \text{Quadratic Convergence Rate.}} \tag{6.49}$$

In contrast, for the linear convergence, the iterations approach the optimal according to:

$$\boxed{\ln \frac{1}{||\boldsymbol{\theta}^{(i)} - \boldsymbol{\theta}_*||^2} \propto i: \quad \text{Linear Convergence Rate.}} \tag{6.50}$$

Furthermore, the presence of the Hessian in the correction term remedies, to a large extent, the influence of the condition number of the Hessian matrix on the convergence [6] (Problem 6.16).

6.7.1 RLS AND NEWTON'S METHOD

The RLS algorithm can be rederived following Newton's iterative scheme applied to the MSE and adopting *stochastic approximation* arguments. Let

$$J(\boldsymbol{\theta}) = \frac{1}{2}\,\mathbb{E}\left[(\mathrm{y} - \boldsymbol{\theta}^{\mathrm{T}}\mathbf{x})^2\right] = \frac{1}{2}\sigma_y^2 + \frac{1}{2}\boldsymbol{\theta}^{\mathrm{T}}\Sigma_x\boldsymbol{\theta} - \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{p},$$

or

$$-\nabla J(\boldsymbol{\theta}) = [\boldsymbol{p} - \Sigma_x\boldsymbol{\theta}] = \mathbb{E}[\mathbf{x}\mathrm{e}],$$

and

$$\nabla^2 J(\boldsymbol{\theta}) = \Sigma_x.$$

Newton's iteration becomes

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} + \mu_i\Sigma_x^{-1}\,\mathbb{E}[\mathbf{x}\mathrm{e}].$$

Following stochastic approximation arguments and replacing iteration steps with time updates and expectations with observations, we obtain

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + \mu_n\Sigma_x^{-1}\boldsymbol{x}_n e_n.$$

Let us now adopt the approximation,

$$\Sigma_x \simeq \frac{1}{n+1}\Phi_n = \left(\frac{1}{n+1}\lambda\beta^{n+1}I + \frac{1}{n+1}\sum_{i=0}^{n}\beta^{n-i}\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}\right),$$

and set

$$\mu_n = \frac{1}{n+1}.$$

Then

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + K_n e_n,$$

with

$$K_n = P_n\boldsymbol{x}_n,$$

where

$$P_n = \left(\sum_{i=0}^{n}\beta^{n-i}\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}} + \lambda\beta^{n+1}I\right)^{-1},$$

which then, by using similar steps as for (6.40)–(6.42) leads to the RLS scheme.

Note that this point of view justifies the fast converging properties of the RLS and its relative insensitivity to the condition number of the covariance matrix.

Remarks 6.3.

- When dealing with the LMS in Chapter 5, we saw that LMS is optimal with respect to a min/max robustness criterion. However, this is not true for the RLS. It turns out that while LMS exhibits the best worst case performance, the RLS is expected to have better performance on average [23].

6.8 STEADY-STATE PERFORMANCE OF THE RLS

Compared to the stochastic gradient techniques, which were considered in Chapter 5, we do not have to worry whether RLS converges and where it converges. The RLS computes the *exact* solution of the minimization task in (6.36) in an iterative way. Asymptotically and for $\beta = 1$ ($\lambda = 0$) solves the MSE optimization task. However, we have to consider its steady state performance for $\beta \neq 1$. Even for the stationary case, $\beta \neq 1$ results in an excess mean-square-error. Moreover, it is important to get a feeling of its tracking performance in time-varying environments. To this end, we adopt the same setting as the one followed in Section 5.12. We will not provide all the details of the proof, because this follows similar steps as in the LMS case. We will point out where differences arise and state the results. For the detailed derivation, the interested reader may consult [15, 48, 57]; in the latter one, the energy conservation theory is employed.

As in Chapter 5, we adopt the following models

$$\mathrm{y}_n = \boldsymbol{\theta}_{o,n-1}^T \mathbf{x}_n + \boldsymbol{\eta}_n, \tag{6.51}$$

and

$$\boldsymbol{\theta}_{o,n} = \boldsymbol{\theta}_{o,n-1} + \boldsymbol{\omega}_n, \tag{6.52}$$

with

$$\mathbb{E}[\boldsymbol{\omega}_n \boldsymbol{\omega}_n^{\mathrm{T}}] = \Sigma_\omega.$$

Hence, taking into account (6.51), (6.52), and the RLS iteration involving the respective random variables, we get

$$\boldsymbol{\theta}_n - \boldsymbol{\theta}_{o,n} = \boldsymbol{\theta}_{n-1} + \mathrm{K}_n \mathrm{e}_n - \boldsymbol{\theta}_{o,n-1} - \boldsymbol{\omega}_n,$$

or

$$\begin{aligned}\mathbf{c}_n := \boldsymbol{\theta}_n - \boldsymbol{\theta}_{o,n} &= \mathbf{c}_{n-1} + \mathrm{P}_n \mathbf{x}_n \mathrm{e}_n - \boldsymbol{\omega}_n \\ &= (I - \mathrm{P}_n \mathbf{x}_n \mathbf{x}_n^{\mathrm{T}}) \mathbf{c}_{n-1} + \mathrm{P}_n \mathbf{x}_n \boldsymbol{\eta}_n - \boldsymbol{\omega}_n,\end{aligned}$$

which is the counterpart of (5.78); Note that the time indexes for the input and the noise variables can be dropped out, because their statistics is assumed to be time invariant.

We adopt the same assumptions as in Section 5.12. In addition, we assume that P_n is changing slowly compared to $\mathbf{c}_n$. Hence, every time P_n appears inside an expectation, it is substituted by its mean $\mathbb{E}[P_n]$, namely,

$$\mathbb{E}[\mathrm{P}_n] = \mathbb{E}[\Phi_n^{-1}],$$

where

$$\Phi_n = \lambda \beta^{n+1} I + \sum_{i=0}^{n} \beta^{n-i} \mathbf{x}_i \mathbf{x}_i^{\mathrm{T}},$$

and

$$\mathbb{E}\left[\Phi_n\right] = \lambda \beta^{n+1} I + \frac{1 - \beta^{n+1}}{1 - \beta} \Sigma_x.$$

Assuming $\beta \simeq 1$, the variance at the steady state of Φ_n can be considered small and we can adopt the following approximation,

Table 6.1 The Steady-State Excess MSE, for Small Values of μ and β

Algorithm	Excess MSE, J_{exc}, at Steady-state
LMS	$\frac{1}{2}\mu\sigma_\eta^2\text{trace}\{\Sigma_x\} + \frac{1}{2}\mu^{-1}\text{trace}\{\Sigma_\omega\}$
APA	$\frac{1}{2}\mu\sigma_\eta^2\text{trace}\{\Sigma_x\}\,\mathbb{E}\left[\frac{q}{\|x\|^2}\right] + \frac{1}{2}\mu^{-1}\text{trace}\{\Sigma_x\}\text{trace}\{\Sigma_\omega\}$
RLS	$\frac{1}{2}(1-\beta)\sigma_\eta^2 l + \frac{1}{2}(1-\beta)^{-1}\text{trace}\{\Sigma_\omega\Sigma_x\}$

For $q = 1$, the normalized LMS results. Under a Gaussian input assumption and for long system orders, l, in the APA, $\mathbb{E}\left[\frac{q}{\|x\|}\right] \simeq \frac{q}{\sigma_x^2(l-2)}$ [11].

$$\mathbb{E}[\mathrm{P}_n] \simeq [\mathbb{E}[\Phi_n]]^{-1} = \left[\beta^{n+1}\lambda I + \frac{1-\beta^{n+1}}{1-\beta}\Sigma_x\right]^{-1}.$$

Based on all the previously stated assumptions, then repeating carefully the same steps as in Section 5.12, we end up with the result shown in Table 6.1, which holds for small values of β. For comparison reasons, the excess MSE is shown together with the values obtained for the LMS as well as the APA algorithms. In stationary environments, one simply sets $\Sigma_\omega = 0$. According to Table 6.1, the following remarks are in order:

Remarks 6.4.

- For stationary environments, the performance of the RLS is independent of Σ_x. Of course, if one knows that the environment is stationary, then ideally $\beta = 1$ should be the choice. Yet recall that for $\beta = 1$, the algorithm has stability problems.
- Note that for small μ and $\beta \simeq 1$, there is an "equivalence" of $\mu \simeq 1 - \beta$, for the two parameters in the LMS and RLS. That is, larger values of μ are beneficial to the tracking performance of LMS, while smaller values of β are required for faster tracking; this is expected because the algorithm forgets the past.
- It is clear from Table 6.1 that an algorithm may converge to the steady-state quickly, but it may not necessarily track fast. It all depends on the specific scenario. For example, under the modeling assumptions associated with Table 6.1, the optimal value μ_{opt} for the LMS (Section 5.12) is given by

$$\mu_{\text{opt}} = \sqrt{\frac{\text{trace}\{\Sigma_\omega\}}{\sigma_\eta^2\text{trace}\{\Sigma_x\}}},$$

which corresponds to

$$J_{\text{min}}^{LMS} = \sqrt{\sigma_\eta^2\text{trace}\{\Sigma_x\}\text{trace}\{\Sigma_\omega\}}.$$

Optimizing with respect to β for the RLS, it is easily shown that

$$\beta_{\text{opt}} = 1 - \sqrt{\frac{\text{trace}\{\Sigma_\omega\Sigma_x\}}{\sigma_\eta^2 l}},$$

$$J_{\text{min}}^{\text{RLS}} = \sqrt{\sigma_\eta^2 l\,\text{trace}\{\Sigma_\omega\Sigma_x\}}.$$

Hence, the ratio

$$\boxed{\frac{J_{\min}^{\text{LMS}}}{J_{\min}^{\text{RLS}}} = \sqrt{\frac{\text{trace}\{\Sigma_x\}\text{trace}\{\Sigma_\omega\}}{l\text{trace}\{\Sigma_\omega \Sigma_x\}}},}$$

depends on Σ_ω and Σ_x. *Sometimes LMS tracks better, yet in other problems RLS is the winner.* Having said that, it must be pointed out that the RLS *always* converges faster, and the difference in the rate, compared to the LMS, increases with the condition number of the input covariance matrix.

6.9 COMPLEX-VALUED DATA: THE WIDELY LINEAR RLS

Following similar arguments as in Section 5.7, let

$$\boldsymbol{\varphi} = \begin{bmatrix} \boldsymbol{\theta} \\ \boldsymbol{v} \end{bmatrix}, \quad \tilde{\boldsymbol{x}}_n = \begin{bmatrix} \boldsymbol{x}_n \\ \boldsymbol{x}_n^* \end{bmatrix},$$

with

$$\hat{y}_n = \boldsymbol{\varphi}^{\text{H}} \tilde{\boldsymbol{x}}_n.$$

The least-squares regularized cost becomes

$$J(\boldsymbol{\varphi}) = \sum_{i=0}^{n} \beta^{n-i} (y_n - \boldsymbol{\varphi}^{\text{H}} \tilde{\boldsymbol{x}}_n)(y_n - \boldsymbol{\varphi}^{\text{H}} \tilde{\boldsymbol{x}}_n)^* + \lambda \beta^{n+1} \boldsymbol{\varphi}^{\text{H}} \boldsymbol{\varphi},$$

or

$$\begin{aligned} J(\boldsymbol{\varphi}) = &\sum_{i=0}^{n} \beta^{n-i} |y_n|^2 + \sum_{i=0}^{n} \beta^{n-i} \boldsymbol{\varphi}^{\text{H}} \tilde{\boldsymbol{x}}_n \tilde{\boldsymbol{x}}_n^{\text{H}} \boldsymbol{\varphi} \\ &- \sum_{i=0}^{n} \beta^{n-i} y_n \tilde{\boldsymbol{x}}_n^{\text{H}} \boldsymbol{\varphi} - \sum_{i=0}^{n} \beta^{n-i} \boldsymbol{\varphi}^{\text{H}} \tilde{\boldsymbol{x}}_n y_n^* + \lambda \beta^{n+1} \boldsymbol{\varphi}^{\text{H}} \boldsymbol{\varphi}. \end{aligned}$$

Taking the gradient with respect to $\boldsymbol{\varphi}^*$ and equating to zero, we obtain

$$\boxed{\tilde{\Phi}_n \boldsymbol{\varphi}_n = \tilde{\boldsymbol{p}}_n : \quad \text{Widely Linear LS Estimate,}} \tag{6.53}$$

where

$$\tilde{\Phi}_n = \beta^{n+1} \lambda I + \sum_{i=0}^{n} \beta^{n-i} \tilde{\boldsymbol{x}}_n \tilde{\boldsymbol{x}}_n^{\text{H}}, \tag{6.54}$$

$$\tilde{\boldsymbol{p}}_n = \sum_{i=0}^{n} \beta^{n-i} \tilde{\boldsymbol{x}}_n y_n^*. \tag{6.55}$$

Following similar steps as for the real-valued RLS, the Algorithm 6.2 results, where $\tilde{P}_n := \tilde{\Phi}^{-1}$.

Algorithm 6.2 .[(The widely linear RLS algorithm)]

- Initialize
  - $\boldsymbol{\varphi}_0 = \mathbf{0}$
  - $\tilde{P}_{-1} = \lambda^{-1} I$
  - Select β
- **For** $n = 0, 1, 2, \ldots,$ **Do**
  - $e_n = y_n - \boldsymbol{\varphi}_{n-1}^{\mathrm{H}} \tilde{\boldsymbol{x}}_n$
  - $\boldsymbol{z}_n = \tilde{P}_{n-1} \tilde{\boldsymbol{x}}_n$
  - $K_n = \frac{\boldsymbol{z}_n}{\beta + \tilde{\boldsymbol{x}}_n^{\mathrm{H}} \boldsymbol{z}_n}$
  - $\boldsymbol{\varphi}_n = \boldsymbol{\varphi}_{n-1} + K_n e_n^*$
  - $\tilde{P}_n = \beta^{-1} \tilde{P}_{n-1} - \beta^{-1} K_n \boldsymbol{z}_n^H$
- **End For**

Setting $\boldsymbol{v}_n = 0$ and replacing $\tilde{\boldsymbol{x}}_n$ with $\boldsymbol{x}_n$ and $\boldsymbol{\varphi}_n$ with $\boldsymbol{\theta}_n$, the linear complex-valued RLS results.

6.10 COMPUTATIONAL ASPECTS OF THE LS SOLUTION

The literature concerning the efficient solution of the least-squares equation as well as the computationally efficient implementation of the RLS is huge. In this section, we will only highlight some of the basic directions that have been followed over the years. Most of the available software packages are implementing such efficient schemes.

A major direction in developing various algorithmic schemes was to cope with the numerical stability issues, as we have already discussed in Remarks 6.2. The main concern is to guarantee the symmetry and positive definiteness of Φ_n. The path followed toward this end is to work with the square root factors of Φ_n.

Cholesky factorization

It is known from linear algebra that every positive definite symmetric matrix, such as Φ_n, accepts the following factorization

$$\Phi_n = L_n L_n^{\mathrm{T}},$$

where L_n is lower-triangular with positive entries along its diagonal. Moreover, this factorization is unique.

Concerning our least-squares task, one focuses on updating the factor L_n, instead of Φ_n, in order to improve numerical stability. Computation of the Cholesky factors can be achieved via a modified version of the Gauss elimination scheme [22].

QR factorization

A better option for computing square factors of a matrix, from a numerical stability point of view, is via the QR decomposition method. To simplify the discussion, let us consider $\beta = 1$ and $\lambda = 0$ (no regularization). Then the positive definite (sample) covariance matrix can be factored as

$$\Phi_n = U_n^{\mathrm{T}} U_n.$$

From linear algebra, [22], we know that the $(n+1)\times l$ matrix U_n can be written as a product

$$U_n = Q_n R_n,$$

where Q_n is a $(n+1)\times(n+1)$ orthogonal matrix and R_n is a $(n+1)\times l$ upper triangular matrix. Note that R_n is related to the Cholesky factor L_n^T. It turns out that working with the QR factors of U_n is preferable, with respect to numerical stability, to working on the Cholesky factorization of Φ_n. QR factorization can be achieved via different paths:

- *Gram-Schmidt* orthogonalization of the input matrix columns. We have seen this path in Chapter 4 while discussing the lattice-ladder algorithm for solving the normal equations for the filtering case. Under the time shift property of the input signal, lattice-ladder-type algorithms have also been developed for the least-squares filtering task [31, 32].
- *Givens rotations*: This has also been a popular line [10, 41, 52, 54].
- *Householder reflections*: This line has been followed in Refs. [53, 55]. The use of Householder reflections leads to a particularly robust scheme from a numerical point of view. Moreover, the scheme presents a high degree of parallelism, which can be exploited appropriately in a parallel processing environment.

A selection of related to QR factorization review papers is given in Ref. [2].

Fast RLS versions

Another line of intense activity, especially in the 1980s, was that of exploiting the special structure associated with the filtering task; that is, the input to the filter comprises the samples from a realization of a random signal/process. Abiding by our adopted notational convention, the input vector will now be denoted as $\boldsymbol{u}$ instead of $\boldsymbol{x}$. Also, for the needs of the discussion we will bring into the notation the order of the filter, m. In this case, the input vectors (regressors), at two successive time instants, share all but two of their components. Indeed, for an mth order system, we have that

$$\boldsymbol{u}_{m,n} = \begin{bmatrix} u_n \\ \vdots \\ u_{n-m+1} \end{bmatrix}, \quad \boldsymbol{u}_{m,n-1} = \begin{bmatrix} u_{n-1} \\ \vdots \\ u_{n-m} \end{bmatrix},$$

and we can partition the input vector as

$$\boldsymbol{u}_{m,n} = \left[u_n \, \boldsymbol{u}_{m-1,n-1}\right]^{\mathrm{T}} = \left[\boldsymbol{u}_{m-1,n}, \, u_{n-m+1}\right]^{\mathrm{T}}.$$

This property is also known as *time-shift* structure. Such a partition of the input vector leads to

$$\begin{aligned} \Phi_{m,n} &= \begin{bmatrix} \sum_{i=0}^{n} u_i^2 & \sum_{i=0}^{n} \boldsymbol{u}_{m-1,i-1}^T u_i \\ \sum_{i=0}^{n} \boldsymbol{u}_{m-1,i-1} u_i & \Phi_{m-1,n-1} \end{bmatrix} \\ &= \begin{bmatrix} \Phi_{m-1,n} & \sum_{i=0}^{n} \boldsymbol{u}_{m-1,i} u_{i-m+1} \\ \sum_{i=0}^{n} \boldsymbol{u}_{m-1,i}^T u_{i-m+1} & \sum_{i=0}^{n} u_{i-m+1}^2 \end{bmatrix}, \; m = 2, 3, \ldots, l \end{aligned} \tag{6.56}$$

where for complex variables transposition is replaced by the Hermitian one. Compare (6.56) with (4.60). The two partitions look alike, yet they are different. Matrix $\Phi_{m,n}$ is no longer Toeplitz. Its low partition is given in terms of $\Phi_{m-1,n-1}$. Such matrices are known as *near-to-Toeplitz*. All that is needed is to "correct" $\Phi_{m-1,n-1}$ back to $\Phi_{m-1,n}$ subtracting a rank one matrix, i.e.,

$$\Phi_{m-1,n-1} = \Phi_{m-1,n} - \boldsymbol{u}_{m-1,n}\boldsymbol{u}_{m-1,n}^{\mathrm{T}}.$$

It turns out that such corrections, although they may slightly complicate the derivation, can still lead to computational efficient order recursive schemes, via the application of the matrix inversion lemma, as was the case in the MSE of Section 4.8. Such schemes have their origin in the pioneering PhD thesis of Martin Morf at Stanford [42]. Levinson-type, Schur-type, split-Levinson type, lattice-ladder algorithms have been derived for the least-squares case [3, 27, 28, 43, 44, 60, 61]. Some of the schemes noted previously under the *QR* factorization exploit the time-shift structure of the input signal.

Besides the order recursive schemes, a number of fixed order fast RLS-type schemes have been developed following the work in Ref. [33]. Recall from the definition of the Kalman gain in (6.43) that for an lth order system we have

$$\begin{aligned} K_{l+1,n} = \Phi_{l+1,n}^{-1}\boldsymbol{u}_{l+1,n} &= \begin{bmatrix} * & * \\ * & \Phi_{l,n-1} \end{bmatrix}^{-1} \begin{bmatrix} * \\ \boldsymbol{u}_{l,n-1} \end{bmatrix} \\ &= \begin{bmatrix} \Phi_{l,n} & * \\ * & * \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{u}_{l,n} \\ * \end{bmatrix}, \end{aligned}$$

where $*$ denotes any value of the element. Without going into detail, the low partition can relate the Kalman gain of order l and time $n-1$ to the Kalman gain of order $l+1$ and time n (step up). Then the upper partition can be used to obtain the time update Kalman gain at order l and time n (step down). Such a procedure bypasses the need for matrix operations leading to $\mathcal{O}(l)$ RLS type algorithms [7, 9] with complexity $7l$ per time update. However, these versions turned out to be numerically unstable. Numerically stabilized versions, at only a small extra computational cost, were proposed in Refs. [5, 58]. All the aforementioned schemes have also been developed for solving the (regularized) exponentially weighted LS cost function.

Besides this line, variants that obtain approximate solutions have been derived in an attempt to reduce complexity; these schemes use an approximation of the covariance or inverse covariance matrix [14, 38]. The *fast Newton transversal filter* (FNTF) algorithm [45], approximates the inverse covariance matrix by a banded matrix of width p. Such a modeling has a specific physical interpretation. A banded inverse covariance matrix corresponds to an AR process of order p. Hence, if the input signal can sufficiently be modeled by an AR model, FNTF obtains a least-squares performance. Moreover, this performance is obtained at $\mathcal{O}(p)$ instead of $\mathcal{O}(l)$ computational cost. This can be very effective in applications where $p \ll l$. This is the case, for example, in audio conferencing, where the input signal is speech. Speech can efficiently be modeled by an AR of the order of 15, yet the filter order can be of a few hundred taps [49]. FNTF bridges the gap between LMS ($p = 1$) and (fast) RLS ($p = l$). Moreover, FNTF builds upon the structure of the stabilized fast RLS. More recently, the banded inverse covariance matrix approximation has been successfully applied in spectral analysis [21].

More on the efficient least-squares schemes can be found in Refs. [15, 17, 20, 24, 29, 57].

6.11 THE COORDINATE AND CYCLIC COORDINATE DESCENT METHODS

So far, we have discussed the steepest descent and Newton's method for optimization. We will conclude the discussion with a third method, which can also be seen as a member of the steepest descent family of methods. Instead of the Euclidean and quadratic norms, let us consider the following minimization task for obtaining the normalized descent direction,

$$\boldsymbol{v} = \arg\min_{z} z^{\mathrm{T}} \nabla J, \tag{6.57}$$

$$\text{s.t.} \quad ||z||_1 = 1, \tag{6.58}$$

where $||\cdot||_1$ denotes the ℓ_1 norm, defined as

$$||z||_1 := \sum_{i=1}^{l} |z_i|.$$

Most of Chapter 9 is dedicated to this norm and its properties. Observe that this is not differentiable. Solving the minimization task (Problem 6.17) results in,

$$\boldsymbol{v} = -\operatorname{sgn}\left((\nabla J)_k\right) \boldsymbol{e}_k,$$

where $\boldsymbol{e}_k$ is the direction of the coordinate corresponding to the component $(\nabla J)_k$ with the largest absolute value, i.e.,

$$|(\nabla J)_k| > |(\nabla J)_j|, \quad j \neq k,$$

and sgn$(\cdot)$ is the sign function. The geometry is illustrated in Figure 6.7. In other words, the descent direction is along a single basis vector; that is, each time only a *single component* of $\boldsymbol{\theta}$ is updated. It is the component that corresponds to the directional derivative, $\left(\nabla J(\boldsymbol{\theta}^{(i-1)})\right)_k$, with the largest increase and the update rule becomes

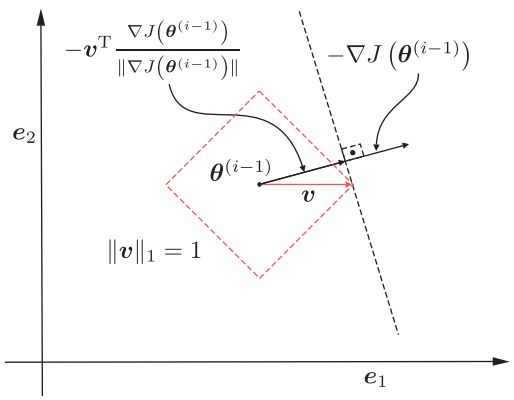

FIGURE 6.7

The unit norm $||\cdot||_1$ ball centered at $\boldsymbol{\theta}^{(i-1)}$ is a rhombus (in $\mathbb{R}^2$). The direction $\boldsymbol{e}_1$ is the one corresponding to the largest component of ∇J. Recall that the components of the vector ∇J are the respective directional derivatives.

$$\theta_k^{(i)} = \theta_k^{(i-1)} - \mu_i \frac{\partial J\left(\boldsymbol{\theta}^{(i-1)}\right)}{\partial \theta_k} : \quad \text{Coordinate-Descent Scheme} \tag{6.59}$$

$$\theta_j^{(i)} = \theta_j^{(i-1)}, \quad j = 1, 2, \ldots, l, \; j \neq k. \tag{6.60}$$

Because only one component is updated at each iteration, this greatly simplifies the update mechanism. The method is known as *Coordinate Descent* (CD).

Based on this rationale, a number of variants of the basic coordinate descent have been proposed. The *Cyclic Coordinate Descent* (CCD) in its simplest form entails a *cyclic* update with respect to one coordinate per iteration cycle; that is, at the ith iteration the following minimization is solved:

$$\theta_k^{(i)} := \arg\min_{\theta} J(\theta_1^{(i)}, \ldots, \theta_{k-1}^{(i)}, \theta, \theta_{k+1}^{(i-1)}, \ldots, \theta_l^{(i-1)}).$$

In words, all components but θ_k are assumed constant; those components θ_j, $j < k$ are fixed to their updated values, $\theta_j^{(i)}$, $j = 1, 2, \ldots, k-1$, and the rest, θ_j, $j = k+1, \ldots, l$, to the available estimates, $\theta_j^{(i-1)}$, from the previous iteration. The nice feature of such a technique is that a simple close form solution for the minimizer may be obtained. A revival of such techniques has happened in the context of sparse learning models (Chapter 10) [18, 67]. Convergence issues of CCD have been considered in Ref. [36, 62]. CCD algorithms for the LS task have also been considered [66] and the references therein. Besides the basic CCD scheme, variants are also available, using different scenarios for the choice of the direction to be updated each time, in order to improve convergence, ranging from a random choice to a change of the coordinate systems, which is known as an *adaptive coordinate descent* scheme [35].

6.12 SIMULATION EXAMPLES

In this section, simulation examples are presented concerning the convergence and tracking performance of the RLS compared to algorithms of the gradient descent family, which have been derived in Chapter 5.

Example 6.1. The focus of this example is to demonstrate the comparative performance, with respect to the convergence rate of the RLS, the NLMS, and the APA algorithms, which have been discussed in Chapter 5. To this end, we generate data according to the regression model

$$y_n = \boldsymbol{\theta}_o^{\mathrm{T}} \boldsymbol{x}_n + \eta_n,$$

where $\boldsymbol{\theta}_o \in \mathbb{R}^{200}$. Its elements are generated randomly according to the normalized Gaussian. The noise samples are i.i.d. generated via the zero mean Gaussian with variance equal to $\sigma_\eta^2 = 0.01$. The elements of the input vector are also i.i.d. generated via the normalized Gaussian. Using the generated samples $(y_n, \boldsymbol{x}_n)$, $n = 0, 1, \ldots$, as the training sequence for all three previously stated algorithms, the convergence curves of Figure 6.8 are obtained. The curves show the squared error in dBs ($10 \log_{10}(e_n^2)$), averaged over 100 different realizations of the experiments, as a function of the time index n. The parameters used for the involved algorithms are: (a) For the NLMS, we used $\mu = 1.2$ and $\delta = 0.001$; (b) for the APA, we used $\mu = 0.2$, $\delta = 0.001$, and $q = 30$, and (c) for the RLS, $\beta = 1$ and $\lambda = 0.1$. The parameters for the NLMS and the APA were chosen so that both algorithms converge to the same error floor. The improved performance of the APA concerning the convergence rate compared to the NLMS is readily seen. However, both algorithms fall short when compared to the RLS. Note that the

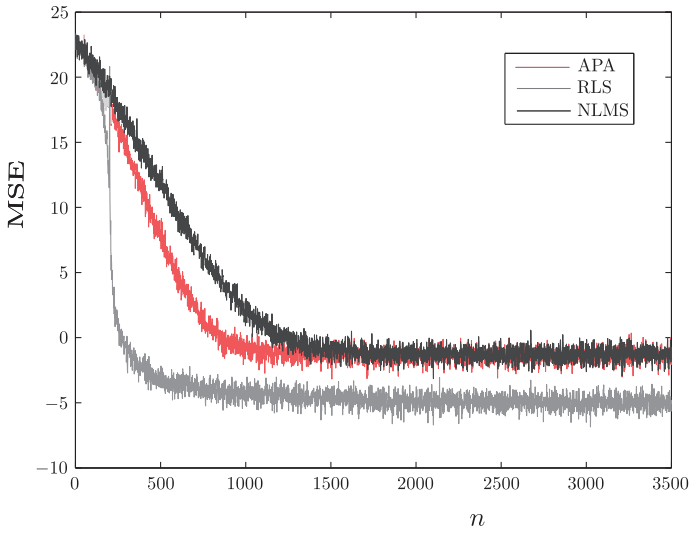

FIGURE 6.8

MSE curves as a function of the number of iterations for NLMS, APA, and RLS. The RLS converges faster and at lower error floor.

RLS converges to lower error floor, because no forgetting factor was used. To be consistent, a forgetting factor $\beta < 1$ should have been used in order for this algorithm to settle at the same error floor as the other two algorithms; this would have a beneficial effect on the convergence rate. However, having chosen $\beta = 1$, is demonstrated that the RLS can converge really fast, even to lower error floors. However, this improved performance is obtained at substantial higher complexity. In case the input vector is part of a random process, and the special time-shift structure can be exploited, as discussed in Section 6.10, the lower complexity versions are at the disposal of the designer. A further comparative performance example, including another family of online algorithms, will be given in Chapter 8.

However, it has to be stressed that this notable advantage (between RLS and LMS-type schemes) in convergence speed, from the initial conditions to steady-state may not be the case concerning the tracking performance, when the algorithms have to track time varying environments. This is demonstrated next.

Example 6.2. This example focuses on the comparative tracking performance of the RLS and NLMS. Our goal is to demonstrate some cases where the RLS fails to do as well as the NLMS. Of course, it must be kept in mind that according to the theory, the comparative performance is very much dependent on the specific application.

For the needs of our example, let us mobilize the time varying model of the parameters given in (6.52) in its more practical version and generate the data according to the following linear system

$$y_n = \boldsymbol{x}_n^T \boldsymbol{\theta}_{o,n-1} + \eta_n, \tag{6.61}$$

where

$$\boldsymbol{\theta}_{o,n} = \alpha \boldsymbol{\theta}_{o,n-1} + \boldsymbol{\omega}_n,$$

with $\boldsymbol{\theta}_{o,n} \in \mathbb{R}^5$. It turns out that such a time varying model is closely related (for the right choice of the involved parameters) to what is known in communications as a Rayleigh fading channel, if the

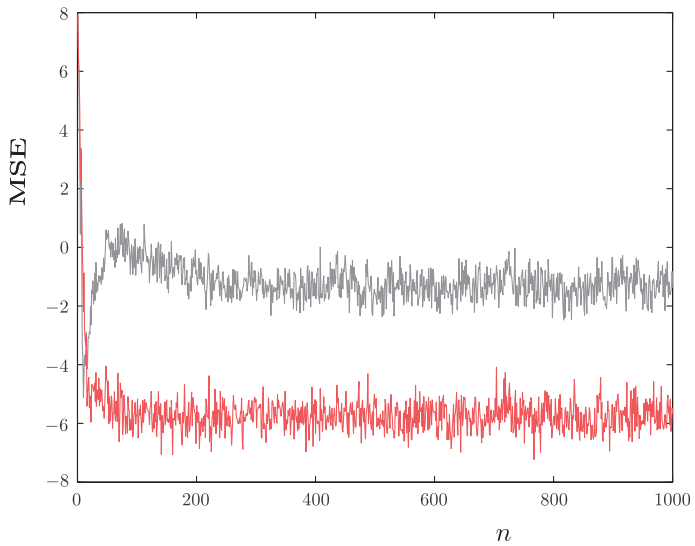

FIGURE 6.9

For a fast time varying parameter model, the RLS (gray) fails to track it, in spite of its very fast initial convergence, compared to the NLMS (red).

parameters comprising $\boldsymbol{\theta}_{o,n}$ are thought to represent the impulse response of such a channel [57]. Rayleigh fading channels are very common and can adequately model a number of transmission channels in wireless communications. Playing with the parameter α and the variance of the corresponding noise source, $\boldsymbol{\omega}$, one can achieve fast or slow time varying scenarios. In our case, we chose $\alpha = 0.97$ and the noise followed a Gaussian distribution of zero mean and covariance matrix $\Sigma_\omega = 0.1I$.

Concerning the data generation, the input samples were generated i.i.d. from a Gaussian $\mathcal{N}(0, 1)$, and the noise was also Gaussian of zero mean value and variance equal to $\sigma_\eta^2 = 0.01$. Initialization of the time varying model ($\boldsymbol{\theta}_{o,0}$) was randomly done by drawing samples from a $\mathcal{N}(0, 1)$.

Figure 6.9 shows the obtained MSE curve as a function of the iterations, for the NLMS and the RLS. For the RLS, the forgetting factor was set equal to $\beta = 0.995$ and for the NLMS, $\mu = 0.5$ and $\delta = 0.001$. Such a choice resulted in the best performance, for both algorithms, after extensive experimentation. The curves are the result of averaging out 200 independent runs.

Figure 6.10 shows the resulting curves for medium and slow time varying channels, corresponding to $\Sigma_\omega = 0.01I$ and $\Sigma_\omega = 0.001I$, respectively.

6.13 TOTAL-LEAST-SQUARES

In this section, the least-squares optimization task will be formulated from a different perspective. Assume zero mean (centered) data and our familiar linear regression model, employing the observed samples,

$$\boldsymbol{y} = X\boldsymbol{\theta} + \boldsymbol{\eta},$$

as in Section 6.2. We have seen that the least-squares task is equivalent with (orthogonally) projecting $\boldsymbol{y}$ onto the span$\{\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_l^c\}$ of the columns of X, hence, making the error

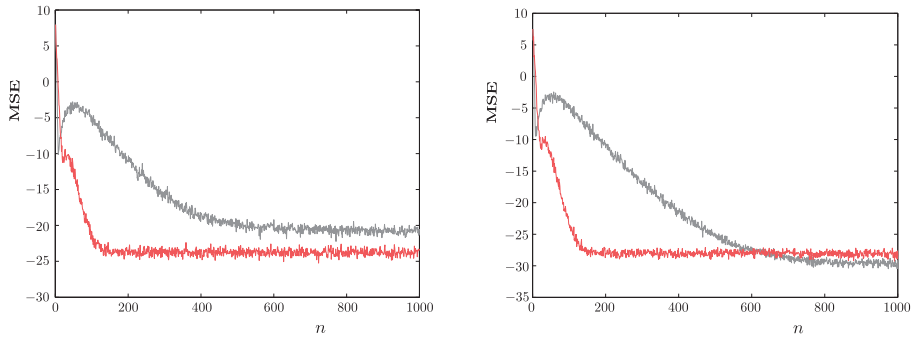

FIGURE 6.10

MSE curves as a function of iteration for (a) a medium and (b) a slow time varying parameter model. The red curve corresponds to the NLMS and the gray one to the RLS.

$$\boldsymbol{e} = \boldsymbol{y} - \hat{\boldsymbol{y}}$$

orthogonal to the column space of X. Equivalently, this can be written as

$$\begin{aligned} &\text{minimize} \quad \|\boldsymbol{e}\|^2, \\ &\text{s.t.} \quad \boldsymbol{y} - \boldsymbol{e} \in \mathcal{R}(X), \end{aligned} \tag{6.62}$$

where $\mathcal{R}(X)$ is the range space of X (see Remarks 6.1 for the respective definition). Moreover, once $\hat{\boldsymbol{\theta}}_{\text{LS}}$ has been obtained, we can write that

$$\hat{\boldsymbol{y}} = X\hat{\boldsymbol{\theta}}_{\text{LS}} = \boldsymbol{y} - \boldsymbol{e}$$

or

$$[X \vdots \boldsymbol{y} - \boldsymbol{e}] \begin{bmatrix} \hat{\boldsymbol{\theta}}_{\text{LS}} \\ -1 \end{bmatrix} = \boldsymbol{0}, \tag{6.63}$$

where $[X \vdots \boldsymbol{y} - \boldsymbol{e}]$ is the matrix that results after extending X by an extra column, $\boldsymbol{y} - \boldsymbol{e}$. Thus, all the points $(y_n - e_n, \boldsymbol{x}_n) \in \mathbb{R}^{l+1}$, $n = 1, 2, \ldots, N$, lie on the same hyperplane, crossing the origin, as shown in Figure 6.11. In other words, in order to fit a hyperplane to the data, the LS method applies a correction e_n, $n = 1, 2, \ldots, N$, to the *output samples*. Thus, we have silently assumed that the regressors have been obtained via exact measurements and that the noise affects *only* the output observations.

In this section, the more general case will be considered, where we allow for both the input (regressors) as well as the output variables to be perturbed by (unobserved) noise samples. Such a treatment has a long history, dating back to the nineteenth century [1]. The method remained in obscurity until it was revived 50 years later for two-dimensional models by Deming [13] and it is sometimes known as *Deming regression*; see also [19] for a historical overview. Such models are also known as *errors-in-variables* regression models.

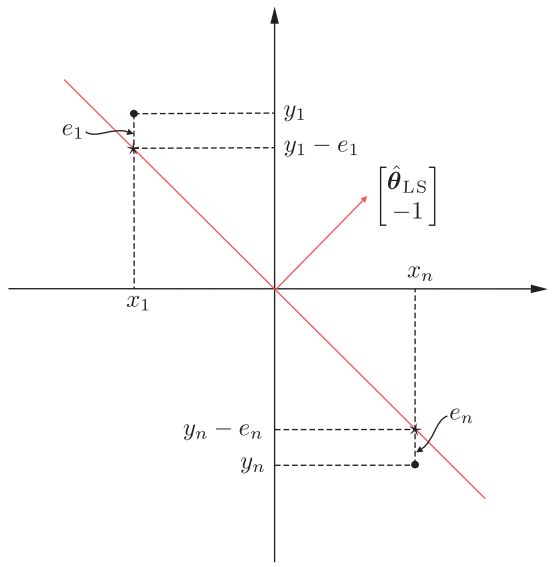

FIGURE 6.11

According to the least-squares method, *only* the output points y_n are corrected to $y_n - e_n$, so that the pairs $(y_n - e_n, x_n)$ lie on a hyperplane, crossing the origin for centered data. If the data are not centered, it crosses the centroid, $(\bar{y}, \bar{x})$.

Our kickoff point is the formulation in (6.62). Let $\boldsymbol{e}$ be the correction vector to be applied on $\boldsymbol{y}$ and E the correction matrix to be applied on X. The method of *total-least-squares* computes the unknown parameter vector by solving the following optimization task,

$$\begin{aligned} \text{minimize} \quad & \|[E \vdots \boldsymbol{e}]\|_F, \\ \text{s.t.} \quad & \boldsymbol{y} - \boldsymbol{e} \in \mathcal{R}(X - E). \end{aligned} \tag{6.64}$$

Recall (Remarks 6.1) that the Frobenius norm of a matrix is defined as the square root of the sum of squares of all its entries, and it is the direct generalization of the Euclidean norm defined for vectors. Let us first focus on solving the task in (6.64), and we will comment on its geometric interpretation, later on. The set of constraints in (6.64) can equivalently be written as

$$(X - E)\boldsymbol{\theta} = \boldsymbol{y} - \boldsymbol{e}. \tag{6.65}$$

Define

$$F := X - E, \tag{6.66}$$

and let $\boldsymbol{f}_i^{\mathrm{T}} \in \mathbb{R}^l$, $i = 1, 2, \ldots, N$, be the rows of F, i.e.,

$$F^{\mathrm{T}} = [\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N],$$

and $\boldsymbol{f}_i^c \in \mathbb{R}^N$, $i = 1, 2, \ldots, l$, the respective columns, i.e.,

$$F = [\boldsymbol{f}_1^c, \ldots, \boldsymbol{f}_l^c].$$

Let also

$$\boldsymbol{g} := \boldsymbol{y} - \boldsymbol{e}. \tag{6.67}$$

Hence, (6.65) can be written in terms of the columns of F, i.e.,

$$\boxed{\theta_1 \boldsymbol{f}_1^c + \cdots + \theta_l \boldsymbol{f}_l^c - \boldsymbol{g} = 0.} \tag{6.68}$$

Equation (6.68) implies that the $l + 1$ vectors, $\boldsymbol{f}_1^c, \ldots, \boldsymbol{f}_l^c, \boldsymbol{g} \in \mathbb{R}^N$, are linearly dependent, which in turn dictates that

$$\boxed{\text{rank}\{[F \vdots \boldsymbol{g}]\} \leq l.} \tag{6.69}$$

There is a subtle point here. The opposite is not necessarily true; that is, (6.69) does not necessarily imply (6.68). If the rank$\{F\} < l$, there is not, in general, $\boldsymbol{\theta}$ to satisfy (6.68). This can easily be verified, for example, by considering the extreme case where $\boldsymbol{f}_1^c = \boldsymbol{f}_2^c = \cdots = \boldsymbol{f}_l^c$. Keeping that in mind, we need to impose some extra assumptions.

Assumptions:

1. The $N \times l$ matrix X is full rank. This implies that all its singular values are nonzero, and we can write (recall (6.12))

$$X = \sum_{i=1}^{l} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}},$$

where we have assumed that

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_l > 0. \tag{6.70}$$

2. The $(N \times (l + 1))$ matrix $[X \vdots \boldsymbol{y}]$ is also full rank, hence

$$[X \vdots \boldsymbol{y}] = \sum_{i=1}^{l+1} \bar{\sigma}_i \bar{\boldsymbol{u}}_i \bar{\boldsymbol{v}}_i^{\mathrm{T}}$$

with

$$\bar{\sigma}_1 \geq \bar{\sigma}_2 \geq \ldots \geq \bar{\sigma}_{l+1} > 0. \tag{6.71}$$

3. Assume that

$$\bar{\sigma}_{l+1} < \sigma_l.$$

As we will see soon, this guarantees the existence of a unique solution. If this condition is not valid, still solutions can exist; however, this corresponds to a degenerate case and such solutions have been the subject of study in the related literature [37, 64]. However, we will not deal with such cases here. Note that, in general, it can be shown that $\bar{\sigma}_{l+1} \leq \sigma_l$ [26]. Thus, our assumption demands *strict* inequality.

4. Assume that

$$\bar{\sigma}_l > \bar{\sigma}_{l+1}.$$

This condition will also be used in order to guarantee uniqueness of the solution.

We are now ready to solve the following optimization task:

$$\boxed{\begin{aligned} \underset{F,\boldsymbol{g}}{\text{minimize}} \quad & \|[X \vdots \boldsymbol{y}] - [F \vdots \boldsymbol{g}]\|_F^2, \\ \text{s.t.} \quad & \text{rank}\{[F \vdots \boldsymbol{g}]\} = l. \end{aligned}} \tag{6.72}$$

In words, compute the best, in the Frobenius norm sense, rank l approximation, $[F \vdots \boldsymbol{g}]$, to the (rank $l+1$) matrix $[X \vdots \boldsymbol{y}]$. We know from Remarks 6.1 that

$$[F \vdots \boldsymbol{g}] = \sum_{i=1}^{l} \bar{\sigma}_i \bar{\boldsymbol{u}}_i \bar{\boldsymbol{v}}_i^{\mathrm{T}}, \tag{6.73}$$

and consequently

$$[E \vdots \boldsymbol{e}] = \bar{\sigma}_{l+1} \bar{\boldsymbol{u}}_{l+1} \bar{\boldsymbol{v}}_{l+1}^{\mathrm{T}}, \tag{6.74}$$

with the corresponding Frobenius and spectral norms of the error matrix being equal to

$$\|E \vdots \boldsymbol{e}\|_F = \bar{\sigma}_{l+1} = \|E \vdots \boldsymbol{e}\|_2. \tag{6.75}$$

Note that the above choice is unique, because $\bar{\sigma}_{l+1} < \bar{\sigma}_l$.

So far, we have uniquely solved the task in (6.72). However, we still have to recover the estimate $\hat{\boldsymbol{\theta}}_{\mathrm{TLS}}$, which will satisfy (6.68). In general, the existence of a unique vector cannot be guaranteed from the F and $\boldsymbol{g}$ given in (6.73). Uniqueness is imposed by assumption (3), which guarantees that the rank of F is equal to l.

Indeed, assume that the rank of F is, k, less than l, $k < l$. Let the best (in the Frobenius/spectral norm sense) rank k approximation of X be X_k and $X - X_k = E_k$. We know from Remarks 6.1 that

$$\|E_k\|_F = \sqrt{\sum_{i=k+1}^{l} \sigma_i^2} \geq \sigma_l.$$

Also, because E_k is the perturbation (error) associated with the best approximation, then

$$\|E\|_F \geq \|E_k\|_F$$

or

$$\|E\|_F \geq \sigma_l.$$

However, from (6.75) we have that

$$\bar{\sigma}_{l+1} = \|E \vdots \boldsymbol{e}\|_F \geq \|E\|_F \geq \sigma_l,$$

which violates assumption (3). Thus, rank$\{F\} = l$. Hence, there is a *unique* $\hat{\boldsymbol{\theta}}_{\text{TLS}}$, such as

$$[F \vdots \boldsymbol{g}] \begin{bmatrix} \hat{\boldsymbol{\theta}}_{\text{TLS}} \\ -1 \end{bmatrix} = \mathbf{0}. \tag{6.76}$$

In other words, $[\hat{\boldsymbol{\theta}}_{\text{TLS}}^T, -1]^T$ belongs to the null space of

$$\begin{bmatrix} F \vdots \boldsymbol{g} \end{bmatrix}, \tag{6.77}$$

which is a rank-deficient matrix; hence, its null space is of dimension one, which is easily checked out that it is spanned by $\bar{\boldsymbol{v}}_{l+1}$, leading to

$$\begin{bmatrix} \hat{\boldsymbol{\theta}}_{\text{TLS}} \\ -1 \end{bmatrix} = \frac{-1}{\bar{v}_{l+1}(l+1)} \bar{\boldsymbol{v}}_{l+1},$$

where $\bar{v}_{l+1}(l+1)$ is the last component of $\bar{\boldsymbol{v}}_{l+1}$. Moreover, it can be shown that (Problem 6.18),

$$\boxed{\hat{\boldsymbol{\theta}}_{\text{TLS}} = (X^{\text{T}}X - \bar{\sigma}_{l+1}^2 I)^{-1} X^{\text{T}}\boldsymbol{y}: \qquad \text{Total-Least-Squares Estimate.}} \tag{6.78}$$

Note that Assumption (3) guarantees that $X^{\text{T}}X - \bar{\sigma}_{l+1}^2 I$ is positive definite (think of why this is so).

Geometric interpretation of the total-least-squares method

From (6.65) and the definition of F in terms of its rows, $\boldsymbol{f}_1^{\text{T}}, \ldots, \boldsymbol{f}_N^{\text{T}}$, we get

$$\boldsymbol{f}_n^{\text{T}} \hat{\boldsymbol{\theta}}_{\text{TLS}} - g_n = 0, \quad n = 1, 2, \ldots, N, \tag{6.79}$$

or

$$\hat{\boldsymbol{\theta}}_{\text{TLS}}^{\text{T}} (\boldsymbol{x}_n - \boldsymbol{e}_n) - (y_n - e_n) = 0. \tag{6.80}$$

In words, both the regressors, $\boldsymbol{x}_n$, as well as the outputs, y_n, are corrected in order for the points, $(y_n - e_n, \boldsymbol{x}_n - \boldsymbol{e}_n)$, $n = 1, 2, \ldots, N$, to lie on a hyperplane in $\mathbb{R}^{l+1}$. Also, once such a hyperplane is computed and it is unique, it has an interesting interpretation. It is the hyperplane that *minimizes the total square distance* of all the training points, $(y_n, \boldsymbol{x}_n)$ from it. Moreover, the corrected points $(y_n - e_n, \boldsymbol{x}_n - \boldsymbol{e}_n) = (g_n, \boldsymbol{f}_n)$, $n = 1, 2, \ldots, N$ are the *orthogonal projections* of the respective $(y_n, \boldsymbol{x}_n)$ training points onto this hyperplane. This is shown in Figure 6.12.

To prove the previous two claims, it suffices to show (Problem 6.19) that the direction of the hyperplane that minimizes the total distance from a set of points, $(y_n, \boldsymbol{x}_n)$, $n = 1, 2, \ldots, N$, is that defined by $\bar{\boldsymbol{v}}_{\boldsymbol{l+1}}$; the latter is the eigenvector associated with the smallest singular value of $[X \vdots \boldsymbol{y}]$, assuming $\bar{\sigma}_l > \bar{\sigma}_{l+1}$.

To see that $(g_n, \boldsymbol{f}_n)$ is the orthogonal projection of $(y_n, \boldsymbol{x}_n)$ on this hyperplane, recall that our task minimizes the following Frobenius norm:

$$||X - F \vdots \boldsymbol{y} - \boldsymbol{g}||_F^2 = \sum_{n=1}^{N} \left((y_n - g_n)^2 + ||\boldsymbol{x}_n - \boldsymbol{f}_n||^2 \right).$$

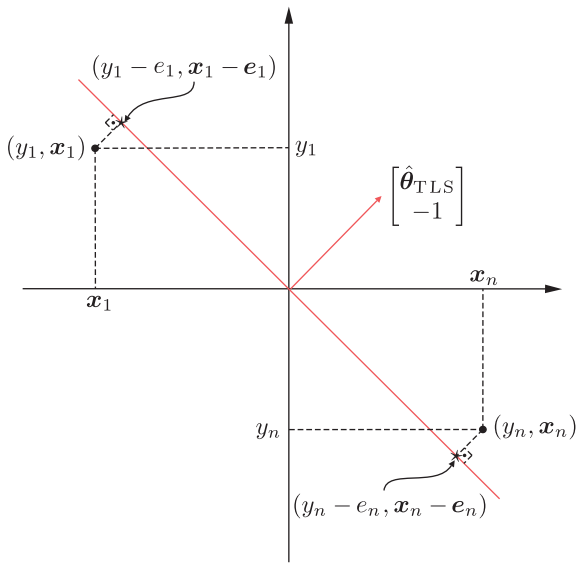

FIGURE 6.12

The total-least-squares method corrects both the values of the output variable as well as the input vector so that the points, after the correction, lie on a hyperplane. The corrected points are the orthogonal projections of $(y_n, \boldsymbol{x}_n)$ on the respective hyperplane; for centered data, this crosses the origin. For noncentered data, it crosses the centroid $(\bar{y}, \bar{\boldsymbol{x}})$.

However, each one of the terms in the above summation is the Euclidean distance between the points $(y_n, \boldsymbol{x}_n)$ and $(g_n, \boldsymbol{f}_n)$, which is minimized if the latter is the orthogonal projection of the former on the hyperplane.

Remarks 6.5.

- The hyperplane defined by the TLS solution,

$$\left[\hat{\boldsymbol{\theta}}_{\mathrm{TLS}}^{T}, -1\right]^{T}$$

minimizes the total distance of all the points $(y_n, \boldsymbol{x}_n)$ from it. We know from geometry that, the squared distance of each point from this hyperplane, is given by

$$\frac{|\hat{\boldsymbol{\theta}}_{\mathrm{TLS}}^{\mathrm{T}} \boldsymbol{x}_n - y_n|^2}{||\boldsymbol{\theta}_{\mathrm{TLS}}^{\mathrm{T}}||^2 + 1}.$$

Thus, $\hat{\boldsymbol{\theta}}_{\mathrm{TLS}}$ minimizes the following ratio:

$$\hat{\boldsymbol{\theta}}_{\mathrm{TLS}} = \arg\min_{\boldsymbol{\theta}} \frac{||X\boldsymbol{\theta} - \boldsymbol{y}||^2}{||\boldsymbol{\theta}||^2 + 1}.$$

This is basically a normalized (weighted) version of the least-squares cost. Looking at it more carefully, TLS promotes vectors of larger norm. This could be seen as a "deregularizing" tendency

of the TLS. From a numerical point of view, this can also be verified by (6.78). The matrix to be inverted for the TLS solution is more ill-conditioned than its LS counterpart. Robustness of the TLS can be improved via the use of regularization. Furthermore, extensions of TLS that employ other cost functions, in order to address the presence of outliers, have also been proposed.

- The TLS method has also been extended to deal with the more general case where $\boldsymbol{y}$ and $\boldsymbol{\theta}$ become matrices. For further reading, the interested reader can look at [37, 64] and the references therein. A distributed algorithm for solving the total-least-squares task in Ad Hoc sensor networks has been proposed in Ref. [4]. A recursive scheme for the efficient solution of the TLS task has appeared in Ref. [12].
- TLS has widely been used in a number of applications, such as computer vision [47]; system identification [59]; speech and image processing [25], [51]; and spectral analysis [63].

Example 6.3. To demonstrate the potential of the total-least-squares to improve upon the performance of the least-squares estimator, in this example, we use noise not only in the input but also in the output samples. To this end, we generate randomly an input matrix, $X \in \mathbb{R}^{150\times 90}$, filling it with elements according to the normalized Gaussian, $\mathcal{N}(0, 1)$. In the sequel, we generate the vector $\boldsymbol{\theta}_o \in \mathbb{R}^{90}$ by randomly drawing samples also from the normalized Gaussian. The output vector is formed as

$$\boldsymbol{y} = X\boldsymbol{\theta}_o.$$

Then, we generate a noise vector $\boldsymbol{\eta} \in \mathbb{R}^{150}$, filling it with elements randomly drawn from $\mathcal{N}(0, 0.01)$, and form

$$\tilde{\boldsymbol{y}} = \boldsymbol{y} + \boldsymbol{\eta}.$$

A noisy version of the input matrix is obtained as

$$\tilde{X} = X + E,$$

where E is filled with elements randomly by drawing samples from $\mathcal{N}(0, 0.2)$.

Using the generated $\tilde{\boldsymbol{y}}$, X, $\tilde{X}$ and pretending that we do not know $\boldsymbol{\theta}_o$, the following three estimates are obtained for its value.

- Using the LS estimator (6.5) together with X and $\tilde{y}$, the average (over 10 different realizations) Euclidean distance of the obtained estimate, $\hat{\boldsymbol{\theta}}$, from the true one is equal to $\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_o\| = 0.0125$.
- Using the LS estimator (6.5) together with $\tilde{X}$ and $\tilde{y}$, the average (over 10 different realizations) Euclidean distance of the obtained estimate $\hat{\boldsymbol{\theta}}$ from the true one is equal to $\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_o\| = 0.4272$.
- Using the TLS estimator (6.78) together with $\tilde{X}$ and $\tilde{y}$, the average (over 10 different realizations) Euclidean distance of the obtained estimate $\hat{\boldsymbol{\theta}}$ from the true one is equal to $\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_o\| = 0.2652$.

Observe that using noisy input data, the LS estimator resulted in higher error compared to the TLS one. Note, however, that the successful application of the TLS presupposes that the assumptions that led to the TLS estimator are valid.

PROBLEMS

6.1 Show that if $A \in \mathbb{C}^{m\times m}$ is positive semidefinite, its trace is nonnegative.

6.2 Show that under (a) the independence assumption of successive observation vectors and (b) the presence of white noise independent of the input, the LS estimator is asymptotically distributed according to the normal distribution, i.e.,

$$\sqrt{N}(\theta - \boldsymbol{\theta}_0) \longrightarrow \mathcal{N}(\mathbf{0}, \sigma^2 \Sigma_x^{-1}),$$

where σ^2 is the noise variance and Σ_x the covariance matrix of the input observation vectors, assuming that it is invertible.

6.3 Let $X \in \mathbb{C}^{m \times l}$. Then show that the two matrices,

$$XX^{\mathrm{H}} \text{ and } X^{\mathrm{H}}X,$$

have the same eigenvalues.

6.4 Show that if $X \in \mathbb{C}^{m \times l}$, then the eigenvalues of XX^{H} ($X^{\mathrm{H}}X$) are real and nonnegative. Moreover, show that if $\lambda_i \neq \lambda_j$, $\boldsymbol{v}_i \perp \boldsymbol{v}_j$.

6.5 Let $X \in \mathbb{C}^{m \times l}$. Then show that if $\boldsymbol{v}_i$ is the normalized eigenvector of $X^{\mathrm{H}}X$, corresponding to λ_i, then the corresponding normalized eigenvector $\boldsymbol{u}_i$ of XX^{H} is given by

$$\boldsymbol{u}_i = \frac{1}{\sqrt{\lambda_i}} X \boldsymbol{v}_i.$$

6.6 Show Eq. (6.17).

6.7 Show that the eigenvectors, $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r$, corresponding to the r singular values of a rank-r matrix, X, solve the following iterative optimization task: compute $\boldsymbol{v}_k$, $k = 2, 3, \ldots, r$, such as,

$$\begin{aligned} &\text{minimize} && \frac{1}{2}||X\boldsymbol{v}||^2, \\ &\text{subject to} && ||\boldsymbol{v}||^2 = 1, \\ & && \boldsymbol{v} \perp \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{k-1}\},\ k \neq 1, \end{aligned}$$

where $||\cdot||$ denotes the Euclidean norm.

6.8 Show that projecting the rows of X onto the k-rank subspace, $V_k = \text{span}\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k\}$, results in the largest variance, compared to any other k-dimensional subspace, Z_k.

6.9 Show that the squared Frobenius norm is equal to the sum of the squared singular values.

6.10 Show that the best k rank approximation of a matrix X or rank $r > k$, in the Frobenius norm sense, is given by

$$\hat{X} = \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T,$$

where σ_i are the singular values and $\boldsymbol{v}_i$, $\boldsymbol{u}_i$, $i = 1, 2, \ldots, r$, are the right and left singular vectors of X, respectively. Then show that the approximation error is given by

$$\sqrt{\sum_{i=k+1}^{r} \sigma_i^2}.$$

6.11 Show that $\hat{X}$, as given in Problem 6.10, also minimizes the spectral norm and that

$$||X - \hat{X}||_2 = \sigma_{k+1}.$$

6.12 Show that the Frobenius and spectral norms are unaffected by multiplication with orthogonal matrices, i.e.,

$$\|X\|_F = \|QXU\|_F$$

and

$$\|X\|_2 = \|QXU\|_2,$$

if $QQ^{\mathrm{T}} = UU^{\mathrm{T}} = I$.

6.13 Show that the null and range spaces of an $m \times l$ matrix, X, of rank r are given by

$$\mathcal{N}(X) = \text{span}\{\boldsymbol{v}_{r+1}, \ldots, \boldsymbol{v}_l\},$$
$$\mathcal{R}(X) = \text{span}\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r\},$$

where

$$X = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m] \begin{bmatrix} D & O \\ O & O \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1^T \\ \vdots \\ \boldsymbol{v}_l^T \end{bmatrix}.$$

6.14 Show that for the ridge regression

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{l} \frac{\sigma_i^2}{\lambda + \sigma_i^2} (\boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{y}) \boldsymbol{u}_i.$$

6.15 Show that the normalized steepest descent direction of $J(\boldsymbol{\theta})$ at a point $\boldsymbol{\theta}_0$ for the quadratic norm $\|v\|_P$ is given by

$$\boldsymbol{v} = -\frac{1}{||P^{-1}\nabla J(\boldsymbol{\theta}_0)||_P} P^{-1} \nabla J(\boldsymbol{\theta}_0).$$

6.16 Justify why the convergence of Newton's iterative minimization method is relatively insensitive on the Hessian matrix.

Hint. Let P be a positive definite matrix. Define a change of variables,

$$\tilde{\boldsymbol{\theta}} = P^{\frac{1}{2}} \boldsymbol{\theta},$$

and carry gradient descent minimization based on the new variable.

6.17 Show that the steepest descent direction, $\boldsymbol{v}$, of $J(\boldsymbol{\theta})$ at a point, $\boldsymbol{\theta}_0$, constrained to

$$\|\boldsymbol{v}\|_1 = 1,$$

is given by $\boldsymbol{e}_k$, where $\boldsymbol{e}_k$ is the standard basis vector in the direction, k, such that

$$|(\nabla J(\boldsymbol{\theta}_0))_k| > |(\nabla J(\boldsymbol{\theta}_0))_j|, \quad k \neq j.$$

6.18 Show that the TLS solution is given by

$$\hat{\boldsymbol{\theta}} = \left(X^{\mathrm{T}} X - \bar{\sigma}_{l+1}^2 I\right)^{-1} X^{\mathrm{T}} \boldsymbol{y},$$

where $\bar{\sigma}_{l+1}$ is the smallest singular value of $[X \vdots \boldsymbol{y}]$.

6.19 Given a set of centered data points, $(y_n, \boldsymbol{x}_n) \in \mathbb{R}^{l+1}$, derive a hyperplane

$$\boldsymbol{a}^T\boldsymbol{x} + y = 0,$$

which crosses the origin, such as the total square distance of all the points from it to be minimum.

MATLAB Exercises

6.20 Consider the regression model,

$$y_n = \boldsymbol{\theta}_o^{\mathrm{T}}\boldsymbol{x}_n + \eta_n,$$

where $\boldsymbol{\theta}_o \in \mathbb{R}^{200}$ ($l = 200$) and the coefficients of the unknown vector are obtained randomly via the Gaussian distribution $\mathcal{N}(0, 1)$. The noise samples are also i.i.d., according to a Gaussian of zero mean and variance $\sigma_\eta^2 = 0.01$. The input sequence is a white noise one, i.i.d. generated via the Gaussian, $\mathcal{N}(0, 1)$.

Using as training data the samples, $(y_n, \boldsymbol{x}_n) \in \mathbb{R} \times \mathbb{R}^{200}$, $n = 1, 2, \ldots$, run the APA (Algorithm 5.2), the NLMS (Algorithm 5.3), and the RLS (Algorithm 6.1) algorithms to estimate the unknown $\boldsymbol{\theta}_o$.

For the APA algorithm, choose $\mu = 0.2$, $\delta = 0.001$, and $q = 30$. Furthermore, in the NLMS set $\mu = 1.2$ and $\delta = 0.001$. Finally, for the RLS set the forgetting factor β equal to 1. Run 100 independent experiments and plot the average error per iteration in dBs, i.e., $10\log_{10}(e_n^2)$, where $e_n^2 = (y_n - \boldsymbol{x}_n^T\boldsymbol{\theta}_{n-1})^2$. Compare the performance of the algorithms.

Keep playing with different parameters and study their effect on the convergence speed and the error floor in which the algorithms converge.

6.21 Consider the linear system

$$y_n = \boldsymbol{x}_n^T\boldsymbol{\theta}_{o,n-1} + \eta_n, \tag{6.81}$$

where $l = 5$ and the unknown vector is time varying. Generate the unknown vector with respect to the following model

$$\boldsymbol{\theta}_{o,n} = \alpha\boldsymbol{\theta}_{o,n-1} + \boldsymbol{\omega}_n,$$

where $\alpha = 0.97$ and the coefficients of $\boldsymbol{\omega}_n$ are i.i.d. drawn from the Gaussian distribution, with zero mean and variance equal to 0.1. Generate the initial value $\boldsymbol{\theta}_{o,0}$ with respect to the $\mathcal{N}(0, 1)$.

The noise samples are i.i.d., having zero mean and variance equal to 0.001. Furthermore, generate the input samples so that they follow the Gaussian distribution $\mathcal{N}(0, 1)$. Compare the performance of the NLMS and RLS algorithms. For the NLMS, set $\mu = 0.5$ and $\delta = 0.001$. For the RLS, set the forgetting factor β equal to 0.995. Run 200 independent experiments and plot the average error per iteration in dBs, i.e., $10\log_{10}(e_n^2)$, with $e_n^2 = (y_n - \boldsymbol{x}_n^{\mathrm{T}}\boldsymbol{\theta}_{n-1})^2$. Compare the performance of the algorithms.

Keep the same parameters, but set the variance associated with $\boldsymbol{\omega}_n$ equal to $0.01,\ 0.001$. Play with different values of the parameters and the variance of the noise $\boldsymbol{\omega}$.

6.22 Generate an 150×90 matrix X, the entries of which follow the Gaussian distribution $\mathcal{N}(0, 1)$. Generate the vector $\boldsymbol{\theta}_o \in \mathbb{R}^{90}$. The coefficients of this vector are i.i.d. obtained, also, via the Gaussian $\mathcal{N}(0, 1)$. Compute the vector $\boldsymbol{y} = X\boldsymbol{\theta}_o$. Add a 90×1 noise vector, $\boldsymbol{\eta}$, to $\boldsymbol{y}$ in order to generate $\tilde{\boldsymbol{y}} = \boldsymbol{y} + \boldsymbol{\eta}$. The elements of $\boldsymbol{\eta}$ are generated via the Gaussian $\mathcal{N}(0, 0.01)$. In the sequel,

add a 150×90 noise matrix, E, so as to produce $\tilde{X} = X + E$; the elements of E are generated according to the Gaussian $\mathcal{N}(0, 0.2)$. Compute the LS estimate, via (6.5), by employing (a) the true input matrix, X, and the noisy output $\tilde{\boldsymbol{y}}$; and (b) the noisy input matrix, $\tilde{X}$, and the noisy output $\tilde{\boldsymbol{y}}$.

In the sequel, compute the TLS estimate via (6.78) using the noisy input matrix, $\tilde{X}$, and the noisy output $\tilde{\boldsymbol{y}}$.

Repeat the experiments a number of times and compute the average Euclidean distances between the obtained estimates for the previous three cases, and the true parameter vector, $\boldsymbol{\theta}_o$.

Play with different noise levels and comment on the results.

REFERENCES

[1] R.J. Adcock, Note on the method of least-squares, Analyst 4(6) (1877) 183-184.

[2] J.A. Apolinario Jr. (Ed.), QRD-RLS Adaptive Filtering, Springer, New York, 2009.

[3] K. Berberidis, S. Theodoridis, Efficient symmetric algorithms for the modified covariance method for autoregressive spectral analysis, IEEE Trans. Signal Process. 41 (1993) 43.

[4] A. Bertrand, M. Moonen, Consensus-based distributed total least-squares estimation in Ad Hoc wireless sensor networks, IEEE Trans. Signal Process. 59 (5) (2011) 2320-2330.

[5] J.L. Botto, G.V. Moustakides, Stabilizing the fast Kalman algorithms, IEEE Trans. Acoust. Speech Signal Process. 37 (1989) 1344-1348.

[6] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.

[7] G. Carayannis, D. Manolakis, N. Kalouptsidis, A fast sequential algorithm for least-squares filtering and prediction, IEEE Trans. Acoust. Speech Signal Process. 31 (1983) 1394-1402.

[8] F.S. Cattivelli, C.G. Lopes, A.H. Sayed, Diffusion recursive least-squares for distributed estimation over adaptive networks, IEEE Trans. Signal Process. 56 (5) (2008) 1865-1877.

[9] J.M. Cioffi, T. Kailath, Fast recursive-least-squares transversal filters for adaptive filtering, IEEE Trans. Acoust. Speech Signal Process. 32 (1984) 304-337.

[10] J.M. Cioffi, The fast adaptive ROTOR's RLS algorithm, IEEE Trans. Acoust. Speech Signal Process. 38 (1990) 631-653.

[11] M.H. Costa, J.C.M. Bermudez, An improved model for the normalized LMS algorithm with Gaussian inputs and large number of coefficients, in: Proceedings, IEEE Conference in Acoustics Speech and Signal Processing, 2002, pp. 1385-1388.

[12] C.E. Davila, An efficient recursive total least-squares algorithm for FIR adaptive filtering, IEEE Trans. Signal Process. 42 (1994) 268-280.

[13] W.E. Deming, Statistical Adjustment of Data, J. Wiley and Sons, 1943.

[14] P.S.R. Diniz, M.L.R. De Campos, A. Antoniou, Analysis of LMS-Newton adaptive filtering algorithms with variable convergence factor, IEEE Trans. Signal Process. 43 (1995) 617-627.

[15] P.S.R. Diniz, Adaptive Filtering: Algorithms and Practical Implementation, third ed., Springer, 2008.

[16] Y.C. Eldar, Minimax MSE estimation of deterministic parameters with noise covariance uncertainties, IEEE Trans. Signal Process. 54 (2006) 138-145.

[17] B. Farhang-Boroujeny, Adaptive Filters: Theory and Applications, J. Wiley, NY, 1999.

[18] J. Friedman, T. Hastie, H. Hofling, R. Tibshirani, Pathwise coordinate optimization, Ann. Appl. Stat. 1 (2007) 302-332.

[19] J.W. Gillard, A historical review of linear regression with errors in both variables, Technical Report, University of Cardiff, School of Mathematics, 2006.

[20] G. Glentis, K. Berberidis, S. Theodoridis, Efficient least-squares adaptive algorithms for FIR transversal filtering, IEEE Signal Process. Mag. 16 (1999) 13-42.
[21] G.O. Glentis, A. Jakobsson, Superfast Approximative Implementation of the IAA Spectral Estimate, IEEE Trans. Signal Process. 60 (1) (2012) 472-478.
[22] G.H. Golub, C.F. Van Loan, Matrix Computations, The Johns Hopkins University Press, 1983.
[23] B. Hassibi, A.H. Sayed, T. Kailath, H^{∞} optimality of the LMS algorithm, IEEE Trans. Signal Process. 44 (1996) 267-280.
[24] S. Haykin, Adaptive Filter Theory, fourth ed., Prentice Hall, NJ, 2002.
[25] K. Hermus, W. Verhelst, P. Lemmerling, P. Wambacq, S. Van Huffel, Perceptual audio modeling with exponentially damped sinusoids, Signal Process. 85 (1) (2005) 163-176.
[26] R.A. Horn, C.R. Johnson, Matrix Analysis, second ed., Cambridge University Press, 2013.
[27] N. Kalouptsidis, G. Carayannis, D. Manolakis, E. Koukoutsis, Efficient recursive in order least-squares FIR filtering and prediction, IEEE Trans. Acoust. Speech Signal Process. 33 (1985) 1175-1187.
[28] N. Kalouptsidis, S. Theodoridis, Parallel implementation of efficient LS algorithms for filtering and prediction, IEEE Trans. Acoust. Speech Signal Process. 35 (1987) 1565-1569.
[29] N. Kalouptsidis, S. Theodoridis, Adaptive System Identification and Signal Processing Algorithms, Prentice Hall, 1993.
[30] A.P. Liavas, P.A. Regalia, On the numerical stability and accuracy of the conventional recursive least-squares algorithm, IEEE Trans. Signal Process. 47 (1999) 88-96.
[31] F. Ling, D. Manolakis, J.G. Proakis, Numerically robust least-squares lattice-ladder algorithms with direct updating of the reflection coefficients, IEEE Trans. Acoust. Speech Signal Process. 34 (1986) 837-845.
[32] D.L. Lee, M. Morf, B. Friedlander, Recursive least-squares ladder estimation algorithms, IEEE Trans. Acoust. Speech Signal Process. 29 (1981) 627-641.
[33] L. Ljung, M. Morf, D. Falconer, Fast calculation of gain matrices for recursive estimation schemes, Int. J. Control 27 (1984) 304-337.
[34] S. Ljung, L. Ljung, Error propagation properties of recursive least-squares adaptation algorithms, Automatica 21 (1985) 157-167.
[35] I. Loshchilov, M. Schoenauer, M. Sebag, Adaptive coordinate descent, in: Proceedings Genetic and Evolutionary Computation Conference (GECCO), ACM Press, 2011, pp. 885-892.
[36] Z. Luo, P. Tseng, On the convergence of the coordinate descent method for convex differentiable minimization, J. Optim. Theory Appl. 72 (1992) 7-35.
[37] I. Markovsky, S. Van Huffel, Overview of total least-squares methods, Signal Process. 87 (10) (2007) 2283-2302.
[38] D.F. Marshall, W.K. Jenkins. A fast quasi-Newton adaptive filtering algorithm, IEEE Trans. Signal Process. 40 (1993) 1652-1662.
[39] G. Mateos, I. Schizas, G.B. Giannakis, Distributed recursive least-squares for consensus-based in-network adaptive estimation, IEEE Trans. Signal Process. 57 (11) (2009) 4583-4588.
[40] G. Mateos, G.B. Giannakis, Distributed recursive least-squares: stability and performance analysis, IEEE Trans. Signal Process. 60 (7) (2012) 3740-3754.
[41] J.G. McWhirter, Recursive least-squares minimization using a systolic array, Proc. SPIE Real Time Signal Process. VI 431 (1983) 105-112.
[42] M. Morf, Fast algorithms for multivariable systems, Ph.D. Thesis, Stanford University, Stanford, CA, 1974.
[43] M. Morf, T. Kailath, Square-root algorithms for least-squares estimation, IEEE Trans. Automat. Control 20 (1975) 487-497.
[44] M. Morf, B. Dickinson, T. Kailath, A. Vieira, Efficient solution of covariance equations for linear prediction, IEEE Trans. Acoust. Speech Signal Process. 25 (1977) 429-433.

[45] G.V. Moustakides, S. Theodoridis, Fast Newton transversal filters: A new class of adaptive estimation algorithms, IEEE Trans. Signal Process. 39 (1991) 2184-2193.
[46] G.V. Moustakides, Study of the transient phase of the forgetting factor RLS, IEEE Trans. Signal Process. 45 (1997) 2468-2476.
[47] M. Mühlich, R. Mester, The role of total least-squares in motion analysis, in: H. Burkhardt (Ed.), Proceedings of the 5th European Conference on Computer Vision, Springer-Verlag, 1998, pp. 305-321.
[48] V.H. Nascimento, M.T.M. Silva, Adaptive filters, in: R. Chellappa, S. Theodoridis (Eds.), Signal Process, E-Ref. 1 (2014) 619-747.
[49] T. Petillon, A. Gilloire, S. Theodoridis, Fast Newton transversal filters: An efficient way for echo cancellation in mobile radio communications, IEEE Trans. Signal Process. 42 (1994) 509-517.
[50] T. Piotrowski, I. Yamada, MV-PURE estimator: Minimum-variance pseudo-unbiased reduced-rank estimator for linearly constrained ill-conditioned inverse problems, IEEE Trans. Signal Process. 56 (2008) 3408-3423.
[51] A. Pruessner, D. O'Leary, Blind deconvolution using a regularized structured total least norm algorithm, SIAM Journal on Matrix Analysis and Applications 24(4) (2003) 1018-1037.
[52] P.A. Regalia, Numerical stability properties of a QR-based fast least-squares algorithm, IEEE Trans. Signal Process. 41 (1993) 2096-2109.
[53] A.A. Rondogiannis, S. Theodoridis, On inverse factorization adaptive least-squares algorithms, Signal Processing 52 (1997) 35-47.
[54] A.A. Rondogiannis, S. Theodoridis, New fast QR decomposition least-squares adaptive algorithms, IEEE Trans. Signal Process. 46 (1998) 2113-2121.
[55] A. Rontogiannis, S. Theodoridis, Householder-Based RLS Algorithms, in: J.A. Apolonario Jr. (Ed.), QRD-RLS Adaptive Filtering, Springer, 2009.
[56] A.H. Sayed, T. Kailath, A state space approach to adaptive RLS filtering, IEEE Signal Processing Magazine 11 (1994) 18-60.
[57] A.H. Sayed, Fundamentals of Adaptive Filtering, J. Wiley Interscience, 2003.
[58] D.T.M. Slock, R. Kailath, Numerically stable fast transversal filters for recursive least-squares adaptive filtering, IEEE Trans. Signal Process. 39 (1991) 92-114.
[59] T. Söderström, Errors-in-variables methods in system identification, Automatica 43(6) (2007) 939-958.
[60] S. Theodoridis, Pipeline architecture for block adaptive LS FIR filtering and prediction, IEEE Trans. Acoust. Speech Signal Process. 38 (1990) 81-90.
[61] S. Theodoridis, A. Liavas, Highly concurrent algorithm for the solution of ρ-Toeplitz system of equations, Signal Process. 24 (1991) 165-176.
[62] P. Tseng, Convergence of a block coordinate descent method for nondifferentiable minimization, J. Optim. Theory Appl. 109 (2001) 475-494.
[63] D. Tufts, R. Kumaresan, Estimation of frequencies of multiple sinusoids: Making linear prediction perform like maximum likelihood, Proc. IEEE 70 (9) (1982) 975-989.
[64] S. Van Huffel, J. Vandewalle, The Total-Least-Squares Problem: Computational Aspects and Analysis, SIAM, Philadelphia, 1991.
[65] M.H. Verhaegen, Round-off error propagation in four generally-applicable, recursive, least-squares estimation schemes, Automatica 25 (1989) 437-444.
[66] G.P. White, Y.V. Zakharov, J. Liu, Low complexity RLS algorithms using dichotomous coordinate descent iterations, IEEE Transactions on Signal Processing 56 (2008) 3150-3161.
[67] T.T. Wu, K. Lange, Coordinate descent algorithms for lasso penalized regression, Ann. Appl. Stat. 2 (2008) 224-244.
[68] B. Yang, A note on the error propagation analysis of recursive least-squares algorithms, IEEE Trans. Signal Process. 42 (1994) 3523-3525.