CHAPTER

Parametric Estimation 11

Suleyman Serdar Kozat* **and Andrew C. Singer†**

*Department of Electrical and Electronics Engineering, Koc University Istanbul, Turkey
†110 CSL, 1308 West Main Street, Urbana, IL, USA

1.11.1 Introduction

This chapter considers the topic of parametric estimation, which is an important engineering concept that is often used for modeling signals and systems. When a complex signal or system is encountered, it is often desirable to construct a model for the signal or system in question such that it can be readily analyzed. For an input-output system, this may be to understand its behavior, its potential weaknesses, or its stability to bounded inputs. For signals of interest, this may be to understand the content of the signal, such as to determine the words that were spoken in an acoustic signal containing a recording of human speech. In many such cases, models for the signals or systems of interest are sought that can be practically analyzed without excess computation. As in Occam's razor, a parsimonious model is better than a needlessly complicated one. A good model would be one that is sufficiently rich to enable insights into the salient characteristics of the signals or systems under study, but not any more so than necessary. By using parametric models, whose behavior is uniquely determined by a finite set of parameters, the complexity of the model can often be controlled by limiting the number of parameters. In this chapter, we focus on the estimation of such model parameters under a variety of scenarios. Using both statistical and deterministic formulations, rational models (linear models with a finite number of poles and zeros) are considered in both the batch and online recursive formulations. Such autoregressive and moving average models form the basis for our initial investigation, after which we explore advanced topics, such as spectrum estimation, prediction, and filtering, as well as a number of nonlinear parametric modeling techniques.

The background assumed in this chapter is a working knowledge of discrete-time signals and systems, some basic probability theory including knowledge of stationary random processes, and the mathematical skills of a senior level undergraduate student in electrical engineering or a similar discipline. An exemplary treatment of these topics includes the texts by Oppenheim et al. [1], Rabiner and Gold [2], and Rabiner and Schafer [3]. These texts cover discrete-time signal and system theory, with the latter covering some random processes. Additional material on probability and stochastic processes as applied to the problems in this chapter can be found in [4,5].

1.11.2 Deterministic and stochastic signals

In this chapter, we will consider discrete-time signals through a number of different lenses with the goal of developing parametric models for signals and systems that are useful in a variety of engineering contexts.

Academic Press Library in Signal Processing. http://dx.doi.org/10.1016/B978-0-12-396502-8.00011-5

In order to proceed, we will often make use of two distinct models for signals in our discussions. The first, and perhaps the simplest formulation, is to assume that the signals of interest are deterministic, that is, that the signal values $x[n]$ can take on any finite real value and there are no probabilities or other constraints to these values. This might be a useful way for us to capture the inability of our modeling methods to provide any other meaningful measure to the data we observe than the simple statement that the data can take on any values.

In this light, we will often use measures of performance such as the accumulated square error of an estimate $\hat{x}[n]$ for a signal $x[n]$ over a block of samples, $n = 0, \ldots, N$,

$$E_{LS} = \sum_{n=0}^{N} (x[n] - \hat{x}[n])^2, \tag{11.1}$$

or we may formulate this as an integral square error measured in the frequency domain,

$$E_{IS} = \frac{1}{2\pi} \int_{-\pi}^{\pi} |X_d(\omega) - \hat{X}(\omega)|^2 d\omega, \tag{11.2}$$

as a means of capturing the degree to which the approximating time domain signal $\hat{x}[n]$ or frequency domain estimate $\widehat{X}(\omega)$ match or model the deterministic signal of interest $x[n]$ or $X_d(\omega)$.

As a simple example, suppose that we have measured a signal $x[n]$ that corresponds to the daily temperature of the water in a local pond. If we wanted to compute a good approximating signal, $\hat{x}[n] = c$, i.e., a signal whose value remains constant over the entire interval, and we use the accumulated square error over the collected data signal as a measure of performance, we could write

$$E_{LS} = \sum_{n=0}^{N} (x[n] - c)^2 \tag{11.3}$$

and seek to find the value of that minimizes the total accumulated square error over the observation interval. Differentiating E_{LS} with respect to c, and setting the result equal to zero, we obtain,

$$\frac{\partial E_{LS}}{\partial c} = -2 \sum_{n=0}^{N} (x[n] - c), \tag{11.4}$$

$$0 = -2 \sum_{n=0}^{N} (x[n] - c), \tag{11.5}$$

which can be solved for the minimizing value of c, yielding

$$c = \frac{1}{N+1} \sum_{n=0}^{N} x[n]. \tag{11.6}$$

This is the well-known result that the sample-average of a sequence is the best approximating value for the sequence, if we restrict the set of possible approximating signals to those that take on only a single constant value.

Figure 11.1 shows the result of approximating a signal $x[n]$ with another signal that takes on only a single constant value. The signal in this example has an average value of c over the interval shown.

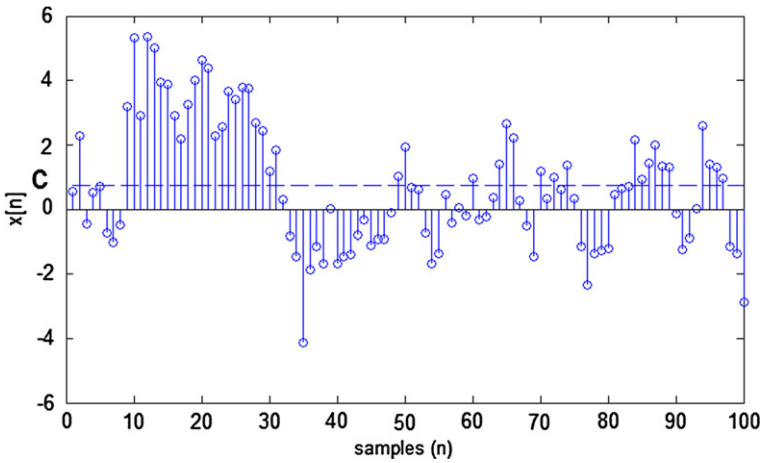

FIGURE 11.1

Approximating a signal with a constant value.

As a result, the best approximating constant valued signal, in terms of minimizing the squared approximation error over the interval shown, is the mean of the signal, c.

Another approach that we will often pursue is the use of stochastic (random) signal models in our development. Random signal models enable the use of probabilistic methods to model some of the uncertainty, or relative uncertainty, in the evolution of a time-series or a given signal or system model. As a result, we may choose measures of performance that are similar in spirit to those of our deterministic signal analyses, but for which we make explicit use of a probabilistic model for the signals of interest. In statistical signal processing, square-error methods are often used, again for their mathematical tractability as well as our ability to capture much of the behavior of the signals of interest with a relatively small number of statistical quantities. Specifically, we will often use a model for random signals that makes use of the so-called second-order statistics of the signal of interest [6]. Specifically, for a discrete-time random signal, $x[n]$, to have a complete statistical characterization of the signal would require knowledge of the probability density function for all possible probabilistic quantities such as $f_x(x[n_1], x[n_2], x[n_3], \ldots, x[n_k])$ for all possible $n_1, n_2, n_3, \ldots, n_k$, and for all possible k. This complete statistical knowledge may be available for certain signal or system models, however, in general such a complete characterization is either not available or not appropriate.

As a result, we will often use second-order statistical models for random signals of interest by assuming that we either have knowledge of, or can make estimates of, the following two quantities. Specifically, the mean-value of the signal is defined as

$$m_x[n] = E\{x[n]\}, \tag{11.7}$$

where the $E\{\,\}$ denotes the statistical expectation operation which is taken with respect to the complete stochastic characterization of the random signal $x[n]$. The covariance function of the signal is defined as

$$\begin{aligned} C_x[m,n] &= E\{(x[m] - m_x[m])(x[n] - m_x[n])^*\}, \\ C_x[m,n] &= E\{(x[m]x[n]*)\} - m_x[m]m_x[n]^*, \end{aligned} \tag{11.8}$$

where, again, the expectation is taken with respect to the complete stochastic characterization of the random signal, and the second term in the definition is conjugated, if the random signal is complex valued.

When the random signal $x[n]$ is *wide-sense stationary* (WSS), then both the mean-value of the signal and also the covariance function take on a special form [1,6]. Specifically, for wide-sense stationary random signals, the mean-value of the signal is not a function of time, i.e., $m_x[n] = m_x$, and the covariance function is only a function of the difference in times between the two samples, i.e.,

$$C_x[m, n] = C_x[0, n - m] \triangleq C_x[n - m], \tag{11.9}$$

where, with a slight abuse of notation, the same name is given to both the general and the wide-sense stationary covariance functions. In general terms, a random signal is stationary if the statistical properties of the signal do not depend on the specific time (of day, of the year, etc.) but rather samples of the signal have correlation (or other statistical dependence) that is merely a function of how close in time the samples are taken with respect to one another.

Returning to our earlier example of representing a signal, now a random signal, by a single constant estimate of that signal, we might seek to minimize the following mean square error, rather than the accumulated square error as before,

$$\begin{aligned} E_{MS} &= E\{(x[n] - \hat{x}[n])^2\}, \\ E_{MS} &= E\{(x[n] - c)^2\}, \end{aligned} \tag{11.10}$$

where, once again we seek a single value of the constant to represent the signal over the region of interest. If the signal $x[n]$ is stationary, then we can find a simple solution to this estimation problem by differentiating with respect to the parameter as follows:

$$\begin{aligned} \frac{\partial E_{MS}}{\partial c} &= -E\{2(x[n] - c)\}, \\ 0 &= -2E\{x[n]\} + 2c, \\ c &= m_x. \end{aligned} \tag{11.11}$$

This provides the stochastic version of the deterministic least squares estimation problem, whereby the minimum mean square error (MMSE) constant estimate of a wide-sense stationary random signal is given by the mean of the signal.

1.11.3 Parametric models for signals and systems

There are a wide variety of ways in which a signal can be represented, or *modeled* using the tools of signal processing and random processes. In this section, we will consider models that are *parametric* in that they contain a number of free parameters that can be adjusted to optimize a given criterion of interest, such as the least square or mean square error metrics described previously. Perhaps the simplest such models are the so-called *autoregressive* (AR) and *moving average* (MA) models that use as free parameters, the coefficients of the numerator and denominator polynomials of a rational transfer function that is driven by a deterministic or random process.

In contrast to such parametric signal models, methods that attempt to model or approximate a given signal without the use of such a global parametric model are often called *non-parametric* signal models. A few examples of non-parametric methods for estimating a signal of interest include estimates of histograms for signal values, piecewise constant or piecewise linear signal models, truncated Fourier series models, and nearest neighbor models. We will not explore such non-parametric methods in this chapter.

1.11.3.1 Autoregressive (AR) models

A discrete-time signal $x_{[n]}$ is an autoregressive (AR) process of order p if it can be written in the form [6]

$$x[n] + a_1 x[n-1] + \cdots + a_p x[n-p] = w[n], \tag{11.12}$$

where $a_1, \ldots, a_p$ are parameters of the AR model and the signal $w[n]$ is a zero mean white noise process of variance σ^2, i.e., we have that $C_w[m] = \sigma^2 \delta[m]$. By rearranging the order of the terms in the definition, we can write the autoregressive process recursively, as

$$x[n] = -a_1 x[n-1] - a_2 x[n-2] - \cdots - a_p x[n-p] + w[n], \tag{11.13}$$

which shows that a pth order autoregressive process can be written as a linear combination of the p most recent values of the signal plus a term that is statistically independent of the past values of the process; this is the so-called innovations sequence, and represents the new information, or the unpredictable components in the random signal (see Figure 11.2).

Another way to view autoregressive signals is by recognizing that the definition of the process also represents a convolution of the input sequence $x[n]$ with the autoregressive parameters, such that the output signal is the innovations sequence, i.e.,

$$\sum_{k=0}^{p} a_k x[n-k] = w[n], \tag{11.14}$$

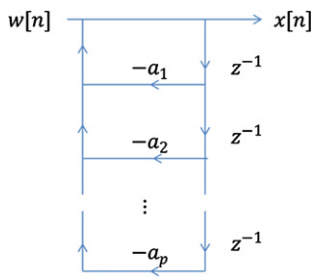

FIGURE 11.2

Autoregressive process flow diagram.

where we set $a_0 = 1$. The significance of this can be seen if we slightly rearrange terms, yielding

$$x[n] - \underbrace{\left(-\sum_{k=1}^{p} a_k x[n-k]\right)}_{\hat{x}[n]} = w[n] \tag{11.15}$$

in which we can view the summation as an estimate of the signal $x[n]$, given by its most recent p parameters, i.e., we have

$$x[n] - \hat{x}[n] = w[n]. \tag{11.16}$$

By writing the process in this form, we see that the sequence $x[n]$ is rather unique in that it can be best estimated by its most recent past values and that after this *predictable component* is subtracted off, all that remains is a white-noise sequence that is independent of the past values. The term "autoregressive" comes from the ability to write the sequence $x[n]$ using a "regression model" over its own past values, i.e., using a finite linear combination of its own past values.

By using the linear regression, or auto-regression model for the sequence $x[n]$, we see that there is an input-output relationship between the sequences $w[n]$ and $x[n]$ that is, we can view the creation of the sequence $x[n]$ by processing the sequence $w[n]$ with a linear, time-invariant filter. Ignoring that the sequences $x[n]$ and $w[n]$ may be random for now, and considering only the input-output relationship between them, we can take the z-transform of both sides (assuming for now that the sequences are deterministic, or given), we have

$$x(z) + \sum_{k=1}^{p} a_k z^{-k} X[z] = W[z], \tag{11.17}$$

$$x(z)\left(1 + \sum_{k=1}^{p} a_k z^{-k}\right) = W[z], \tag{11.18}$$

$$x(z) = \left(\frac{1}{1 + \sum_{k=1}^{p} a_k z^{-k}}\right) W[z]. \tag{11.19}$$

This input-output relationship implies that we can view the sequence $x[n]$ as being generated by filtering the sequence $w[n]$ with an "all-pole" filter, since the transfer function $H(z) = X(z)/W(z)$, relating $X(z)$ and $W(z)$ can be written as a rational transfer function where the numerator polynomial is given by $B(z) = 1$, and the denominator polynomial is given by $A(z) = 1 + \sum_{k=1}^{p} a_k z^{-k}$ (see Figure 11.3).

This input-output interpretation of an AR process provides two interesting ways to view the relationship between the signals $x[n]$ and $w[n]$. Specifically, we can use the relationship $X(z) = H(z)W(z)$,

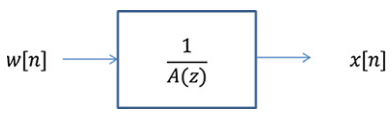

FIGURE 11.3

Input-output transfer function perspective of the autoregressive model.

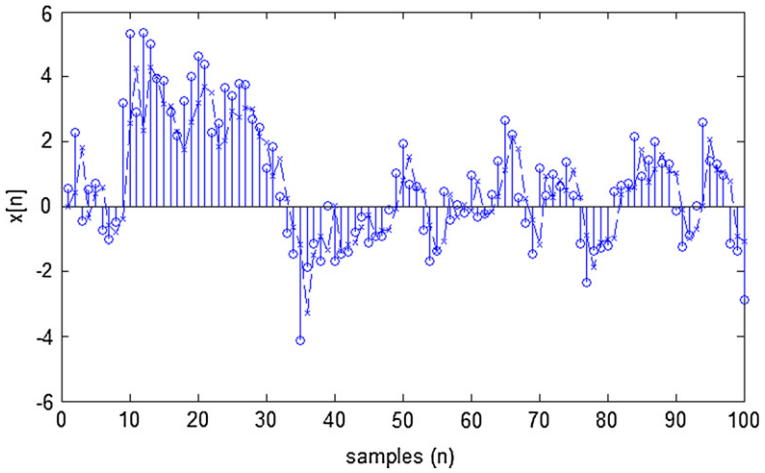

FIGURE 11.4

Tracking colored noise with an AR model.

to look at the signal $x(n)$ as being *generated* by filtering a white-noise signal $w[n]$ with an all-pole filter. This provides us with a simple way to generate random processes (or deterministic signals) with a spectral shape that can be parametrically controlled through the coefficients of $A[z]$. An alternative viewpoint arises from writing the signal in the so-called "prediction error form" [3,6] whereby we have $x[n] - \hat{x}[n] = w[n]$. By writing it this way, we can view the signal $x[n]$ as being filtered using the prediction error filter with transfer function $A(z)$, such that the output is the white-noise signal $w[n]$. In this sense, the predictable component of $x[n]$ can be completely removed by the FIR prediction error filter with transfer function $A(z)$, leaving only the "innovation" sequence, or unpredictable component, $x[n] - \hat{x}[n] = w[n]$.

One application of the AR model is to use the parametric model for prediction of the sequence $x[n]$. For example, one might attempt to predict the next value of a colored noise signal with an AR model. Figure 11.4 shows the input $w[n]$ and output $x[n]$ of an FIR prediction filter that is designed to track the input using only past values of the input. The signal used in Figure 11.1 is reused in Figure 11.4.

Since colored noise can be generated by filtering a white-noise signal, we can view the signal of interest as if it were originally generated by filtering such a white-noise signal. Thus, we expect that the error between the predicted and measured signals will be a white-noise sequence. Figure 11.5 shows the prediction error sequence, which indeed appears uncorrelated from sample to sample, i.e., white.

1.11.3.2 Moving average (MA) models

A discrete-time signal $x[n]$ is called a moving average (MA) process of order q if it can be written in the form [3,6]

$$x[n] = b_0 w[n] + b_1 w[n-1] + \cdots + b_q w[n-q], \tag{11.20}$$

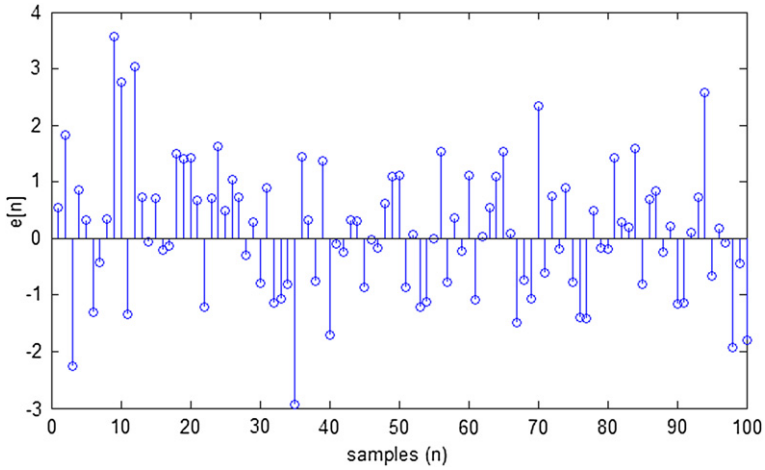

FIGURE 11.5

Prediction error from estimating colored noise using an AR model.

where $b_0, \ldots, b_q$ are parameters of the MA model and the signal $w[n]$ is a white noise process, i.e., we have that $C_w[m] = \sigma^2\delta[m]$. Since the sequence $x[n]$ can be viewed as a linear combination of the most recent values of the sequence then it is called a "moving average." From this structure, whenever the sequence $x[n]$ is given as the output of an FIR filter whose input is a white-noise sequence, then we can refer to as a moving average sequence. More generally, even when the input is not a white-noise sequence, the process of filtering with an FIR filter can be viewed as taking a "moving average" of the input. Financial models often use 50-day and 100-day moving averages as indicators of trends in various stock prices and indices. In such a context, q and b_k are usually taken such that $b_k = \frac{1}{q+1}, k = 0, \ldots, q$, such that $x[n] = \frac{1}{q+1}\sum_{k=0}^{q} w[n-k]$, i.e., the arithmetic mean of the most recent $q+1$ values of $w[n]$.

Viewing the relationship between the sequences $x[n]$ and $w[n]$ with a signal flowgraph model, in Figure 11.6a, we see that $x[n]$ can be viewed as an FIR filtered version of the sequence $w[n]$. Similarly, we can view this in Figure 11.6b. via an input-output transfer function relationship between the two sequences.

1.11.3.3 Autoregressive moving average (ARMA) models

A discrete-time signal $x[n]$ is an autoregressive moving average (ARMA) process of order q, p if it can be written in the form [3,6]

$$x[n] + a_1x[n-1] + \cdots + a_px[n-p] = b_0w[n] + \cdots + b_qw[n-q], \tag{11.21}$$

where $a_1, \ldots, a_p, b_0, \ldots, b_q$ are parameters of the ARMA model and the signal $w[n]$ is a white noise process, i.e., we have that $C_w[m] = \sigma^2\delta[m]$. We can once again relate the sequences $x[n]$ and $w[n]$

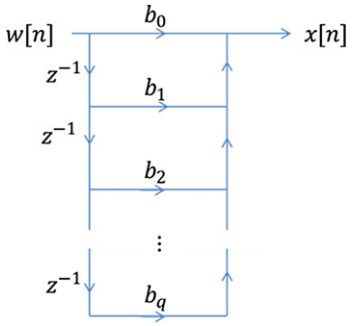

FIGURE 11.6a

Moving average flowgraph model.

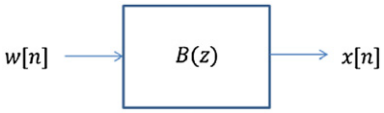

FIGURE 11.6b

Transfer function perspective of the MA model relating $x[n]$ and $w[n]$.

through a transfer function relationship, such as

$$X(z) = \left(\frac{\sum_{k=0}^{q} b_k z^{-k}}{1 + \sum_{k=1}^{p} a_k z^{-k}} \right) W(z). \tag{11.22}$$

This input-output relationship implies that we can view the sequence $x[n]$ as being generated by filtering the sequence $w[n]$ with a rational function, i.e., a "pole-zero" filter with finite order numerator and denominator polynomials, where the numerator polynomial is given by $B(z) = \sum_{k=0}^{q} b_k z^{-k}$, and the denominator polynomial is given by $A(z) = 1 + \sum_{k=1}^{p} a_k z^{-k}$. It is usually assumed that the rational function is "strictly proper," which means that the numerator order is strictly less than that of the denominator, i.e., $q < p$. For this reason, we may refer to the sequence as an ARMA(p) process, rather than ARMA(q, p). Figures 11.7a and 11.7b illustrate the flowgraph and transfer function perspective of the ARMA model relating the sequences $x[n]$ and $w[n]$.

1.11.3.4 Parametric modeling for system function approximation

One of the primary reasons that parametric models are used is to obtain finite-order (rational system) approximations to systems that we encounter in real-world engineering problems. Whether the data of interest are taken from a mechanical system, an electrical device, a digital communications link, or even a financial or economic model, we often seek input-output relationships that can explain the behaviors we observe such that we might be able to better understand them, or even hope to control them.

There are a number of ways in which input-output behavior can be modeled though two are perhaps most common. One is based on observations of input-output pairs in the time domain, from which

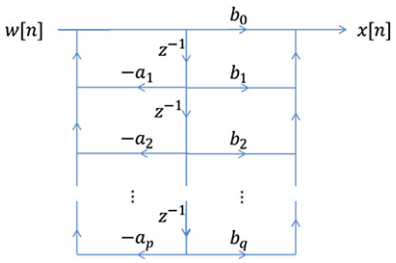

FIGURE 11.7a

ARMA flowgraph model.

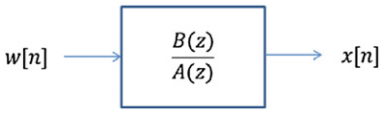

FIGURE 11.7b

Transfer function perspective of the ARMA model relating $x[n]$ and $w[n]$.

an explanation of such behavior is sought. Another other comes from attempts to fit a given system function or frequency response that either arose from a more detailed physical model, or which was measured using instrumentation, such as a spectrum analyzer in response to a variety of probe signals. These two approaches can certainly be linked, through the Fourier transform, and we will move back and forth between the two domains depending on the methods at hand. For example, given a frequency response $H(e^{j\omega})$ that we sought to model, we could readily transform this into the input-output pair $x[n] = \delta[n]$, $y[n] = h[n]$, defining the impulse response of the system as the response of the system to an impulse as input, using the definition of the discrete-time Fourier transform [1],

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h[n]e^{-j\omega n}, \tag{11.23}$$

$$h[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} H(e^{j\omega})e^{j\omega n}\,dw. \tag{11.24}$$

In general, direct ARMA modeling is difficult, since the formulation of the optimization of interest is highly nonlinear in the parameters of the model. For example, given a frequency response $H(e^{j\omega})$, the deterministic least squares optimal ARMA(q, p) model satisfies the following minimization

$$\min_{\{a_k\}_{k=1}^{p}\{b_k\}_{k=0}^{q}} \frac{1}{2\pi}\int_{-\pi}^{\pi}\left|H(e^{j\omega}) - \frac{\sum_{k=0}^{q} b_k e^{-jk\omega}}{1+\sum_{k=1}^{p} a_k e - jk\omega}\right|^2 d\omega, \tag{11.25}$$

which is highly nonlinear in the parameters of the ARMA model. This can be rewritten in the time domain, by application of Pareval's relation [1], which equates energy calculated in the time domain

with that calculated in the frequency domain, yielding,

$$\min_{\{a_k\}_{k=1}^{p}\{b_k\}_{k=0}^{q}} \sum_{n=-\infty}^{\infty} \left| h[n] - \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{\sum_{k=0}^{q} b_k e^{-jk\omega}}{1 + \sum_{k=1}^{p} a_k e - jk\omega} e^{j\omega n} d\omega \right|^2, \tag{11.26}$$

which is equally cumbersome.

This problem formulation is difficult due to the manner in which the AR parameters of interest are nonlinearly related to the optimization criterion. If we were only concerned with a MA estimate and wanted to use this least-squares criterion, then the time-domain formulation can provide a simple solution, whereby

$$\min_{\{b_k\}_{k=0}^{q}} \sum_{n=-\infty}^{\infty} \left| h[n] - \frac{1}{2\pi} \int_{-\pi}^{\pi} \sum_{k=0}^{q} b_k e^{-jk\omega} e^{j\omega n} d\omega \right|^2, \tag{11.27}$$

which can be rewritten

$$\min_{\{b_k\}_{k=0}^{q}} \sum_{n=-\infty}^{\infty} \left| h[n] - \sum_{k=0}^{a} b_k \delta[n-k] \right|^2, \tag{11.28}$$

yielding the simple, though somewhat unsatisfying, solution,

$$b_k = h[k], \quad k = 0, \ldots, q. \tag{11.29}$$

This is nothing more than a simple truncation approximation to the desired impulse response for the system of interest. If a more elaborate response is desired, for example, one that is IIR, then poles are necessary, in addition to the zeros of the approximating transfer function, and an AR or ARMA model should be considered. This again would lead to the nonlinearities we just encountered.

An approach that had proven useful for AR modeling is one that takes the engineering approach to problem solving, that is, if a problem is difficult to solve directly, attempt to change it into one that is more readily solvable. In this light, we will see how the nonlinear optimization for the AR formulation can be transformed into a linear optimization through a clever insight. First, we recall the form of the AR model for the transfer function, namely,

$$H(z) = \frac{1}{1 + \sum_{k=1}^{p} a_k z^{-k}}. \tag{11.30}$$

Beginning with this expression and taking the inverse z-transform, we see that the impulse response of the modeled system must satisfy

$$h[n] + \sum_{k=1}^{p} a_k h[n-k] = \delta[n]. \tag{11.31}$$

Now given a desired (or measured) impulse response, which we will denote, $h_d[n]$, we could attempt to minimize the deviation of this expression from that which is expected from this recursion and replace the nonlinear optimization above with the simpler form

$$\min_{\{a_k\}_{k=1}^{p}} \sum_{n=-\infty}^{\infty} \left| h_d[n] + \sum_{k=1}^{p} a_k h_d[n-k] \right|^2, \tag{11.32}$$

which is a quadratic minimization in the parameters of interest and therefore admits a closed-form solution! Specifically, denoting the quadratic error term in the minimization above as E we can write

$$0 = \frac{\partial E}{\partial a_\ell} = 2 \sum_{n=-\infty}^{\infty} \left(h_d[n] + \sum\nolimits_{k=1}^{p} a_k h_d[n-k] \right) h_d[n-\ell], \quad \ell = 1, \ldots, p, \tag{11.33}$$

which comprise a set of linear equations in a_ℓ, $\ell = 1, \ldots, p$. We can write these equations more compactly by exchanging the order of summation, as

$$\sum_{k=1}^{p} a_k \underbrace{\sum_{n=-\infty}^{\infty} h_d[n-k] h_d[n-\ell]}_{\hat{R}_{h_d h_d}[k-\ell]} = - \underbrace{\sum_{n=-\infty}^{\infty} h_d[n] h_d[n-\ell]}_{\hat{R}_{h_d h_d}[\ell]}, \quad \ell = l, \ldots, p. \tag{11.34}$$

We have made use of the definition which is often called the deterministic auto-correlation sequence for the sequence. We can then summarize the set of equations in matrix form, sometimes referred to as the Normal Equations [3,6],

$$R\vec{a} = -\vec{r}, \tag{11.35}$$

or

$$\begin{bmatrix} \hat{R}_{h_d h_d}[0] & \ldots & \hat{R}_{h_d h_d}[p-1] \\ \vdots & \ddots & \vdots \\ \hat{R}_{h_d h_d}[p-1] & \ldots & \hat{R}_{h_d h_d}[0] \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_p \end{bmatrix} = - \begin{bmatrix} \hat{R}_{h_d h_d}[1] \\ \vdots \\ \hat{R}_{h_d h_d}[p] \end{bmatrix}, \tag{11.36}$$

which has the solution,

$$\vec{a} = -R_{-1}\vec{r}. \tag{11.37}$$

The matrix R has special structure that makes solving such linear equations not only straightforward, but also fast. Namely, R is known as a symmetric Toeplitz matrix, for which fast linear equation solvers can be readily obtained [3,6]. This special structure, together with the right hand side of the linear equations led to a particularly fast algorithm for its solution, known as the Levinson-Durbin recursion [3]. If we were to replace the least-squares enforcement of the prediction error criterion with a mean square error criterion, i.e., we sought to minimize the mean square error

$$\min_{\{a_k\}_{k=1}^{p}} E \left\{ \left| h_d[n] + \sum_{k=1}^{p} a_k h_d[n-k] \right|^2 \right\}, \tag{11.38}$$

we would naturally arrive at a similar form for its solution, namely, we would have

$$R\vec{a} = -\vec{r} \tag{11.39}$$

or

$$\begin{bmatrix} R_{h_d h_d}[0] & \ldots & R_{h_d h_d}[p-1] \\ \vdots & \ddots & \vdots \\ R_{h_d h_d}[p-1] & \ldots & R_{h_d h_d}[0] \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_p \end{bmatrix} = - \begin{bmatrix} R_{h_d h_d}[1] \\ \vdots \\ R_{h_d h_d}[p] \end{bmatrix}, \tag{11.40}$$

which has the solution,

$$\vec{a} = -R^{-1}\vec{r}, \tag{11.41}$$

where, the only difference in our solution lies in replacing the deterministic auto-correlation with the statistical auto-correlation, $R_{h_d h_d}[\ell] = E\{h_d[n]h_d[n-\ell]\}$, where the expectation is taken over the distribution of the random sequence $h_d[n]$. This approach to autoregressive modeling is often called "linear prediction" or "linear predictive modeling" due to the use of the prediction error form that is used to make the optimization problem linear in the parameters of interest [3,6,7].

Returning to the ARMA modeling problem, there are a number of ways of incorporating zeros into the modeling problem while still using the much simpler prediction-error formulation developed so far. Perhaps the simplest is to just use the zeros to match the first values of the impulse response exactly, that is to first solve for the denominator coefficients as above (predictive AR modeling) and then select

$$b_\ell = h_d[\ell] + \sum_{k=1}^{p} a_k h_d[\ell - k], \quad k = 0, \ldots q, \tag{11.42}$$

which follows from the system function relation

$$H_d(z) = \left(\frac{\sum_{k=0}^{q} b_k z^{-k}}{1 + \sum_{k=1}^{p} a_k z^{-k}} \right). \tag{11.43}$$

This two-step process, namely (1) determine the AR parameters using linear predictive modeling, followed by (2) select the MA parameters by exactly matching the first q samples of the impulse response is often called Prony's method [3,7]. In general, this method provides a reasonable set of AR parameters; however the MA parameters are not particularly well-modeled. That is, if a system were known to have a given ARMA form, and its parameters were to be estimated from noisy observations using Prony's method, the AR parameters would likely well-match the true underlying system, however the MA parameters would likely be a poor match.

An improvement over this approach is to once again use a least squares, or MMSE criterion to obtain the MA parameters in a modified two-step approach. The first of the two steps remains the same, that is, use the predictive-least squares (or MMSE) approach to find the AR parameters according to the Normal equations as before. However, now that a set of AR parameters are given, the sequence $v[n]$ is defined as follows:

$$v[n] = -\sum_{k=1}^{p} a_k v[n-k] + \delta[n], \tag{11.44}$$

i.e., $v[n]$ is defined to be the impulse response of the AR model derived in step (1). Now, the goal is to minimize the discrepancy between the desired impulse response $h_d[n]$ and the output of the cascade of the AR model and the MA model, such that the sequence $v[n]$ passed through the FIR filter with transfer function $B(z)$ is close to the desired impulse response. Specifically, we select the MA coefficients to minimize the following criterion,

$$\min_{\{b_k\}_{k=0}^{q}} \sum_{n=-\infty}^{\infty} \left| h_d[n] + \sum_{k=0}^{q} b_k v[n-k] \right|^2, \tag{11.45}$$

which is once again a quadratic minimization admitting a closed-form solution. The linear equations reduce to

$$\sum_{k=0}^{q} b_k \underbrace{\sum_{n=-\infty}^{\infty} v[n-k]v[n-\ell]}_{\hat{R}_{vv}[k-\ell]} = -\underbrace{\sum_{n=-\infty}^{\infty} h_d[n]v[n-\ell]}_{\hat{R}_{h_d v}[\ell]}, \quad \ell = 0, \ldots, q, \tag{11.46}$$

which admits the solution,

$$\vec{b} = -\begin{bmatrix} R_{vv}[0] & \ldots & R_{vv}[q] \\ \vdots & \ddots & \vdots \\ R_{vv}[q] & \ldots & R_{vv}[0] \end{bmatrix}^{-1} \begin{bmatrix} R_{h_d v}[0] \\ \vdots \\ R_{h_d v}[q] \end{bmatrix}, \tag{11.47}$$

$$\vec{b} = -(R_{vv})^{-1}\vec{r}_{hv}, \tag{11.48}$$

where, we denote

$$R_{vv} = \begin{bmatrix} R_{vv}[0] & \ldots & R_{vv}[q] \\ \vdots & \ddots & \vdots \\ R_{vv}[q] & \vdots & R_{vv}[0] \end{bmatrix}, \quad \vec{r}_{hv} = \begin{bmatrix} R_{h_d v}[0] \\ \vdots \\ R_{h_d v}[q] \end{bmatrix}. \tag{11.49}$$

1.11.3.5 Parametric modeling for joint process estimation

One of the classical signal estimation and modeling problems that makes use of parametric estimation methods like those developed here, include the adaptive filtering and joint process estimation problem [2]. Most generally, these problems can be viewed as a parametric modeling problem depicted as follows:

$$x[n] \longrightarrow \boxed{H(e^{j\omega})} \xrightarrow{\hat{d}[n]} \underset{\substack{-\uparrow \\ d[n]}}{\overset{+}{\oplus}} \longrightarrow e[n]$$

where the parameters of a model for $H(e^{j\omega})$ are adjusted to minimize the discrepancy $e[n] = (d[n] - \hat{d}[n])$ in a suitable manner [8]. When the parameters of the model are adjusted in an online (sequential) manner, such that the filter parameters $\{h[k]\}_{k=0}^{p}$ are updated after each observation of the desired signal, this is commonly referred to as "adaptive filtering" [8]. When a batch of data $\{x[k]\}_{k=0}^{N}$ is observed and the parameters $\{h[k]\}_{k=0}^{p}$ are selected based on this entire set of observations, this problem is commonly referred to as batch or block processing. We will first consider the stochastic case, where the discrepancy to be minimized takes the form

$$\{h_{\text{MMSE}}[k]\}_{k=0}^{p} = \arg \min_{\{h[k]\}_{k=0}^{p}} \epsilon\{(d[n] - \hat{d}[n])^2\}. \tag{11.50}$$

Writing out this minimization using the parameters of the model, leads to

$$\{h_{\text{MMSE}}[k]\}_{k=0}^{p} = \arg \min_{\{h[k]\}_{k=0}^{p}} \epsilon_{\text{MMSE}} = E\left\{\left(d[n] - \sum_{k=0}^{p} h[k]x[n-k]\right)^2\right\}. \tag{11.51}$$

To proceed, we consider that the sequences $x[n]$ and $d[n]$ are wide-sense stationary stochastic processes, with zero mean and covariance functions [6]

$$C_x[n, n-m] = E\{x[n]x[n-m]\} = R_x[m] = R_x[-m], \tag{11.52}$$
$$C_d[n, n-m] = E\{d[n]d[n-m]\} = R_d[m] = R_d[-m], \tag{11.53}$$
$$C_{xd}[n, n-m] = E\{x[n]d[n-m]\} = R_{xd}[m] = R_{dx}[-m]. \tag{11.54}$$

We can now solve for the minimizing set of parameters by differentiating this quadratic expression with respect to each of the unknown parameters, which leads to

$$0 = \frac{\partial \epsilon_{\text{MMSE}}}{\partial h[\ell]} = -2E\left\{\left(d[n] - \sum_{k=0}^{p} h[k]x[n-k]\right)x[n-\ell]\right\}, \quad \ell = 0, \ldots, p. \tag{11.55}$$

Rearranging terms, we arrive at

$$0 = E\{d[n]x[n-\ell]\} - \sum_{k=0}^{p} h[k]E\{x[n-k]x[n-\ell]\}, \quad \ell = 0, \ldots, p, \tag{11.56}$$

$$E\{d[n]x[n-\ell]\} = \sum_{k=0}^{p} h[k]E\{x[n-k]x[n-\ell]\}, \quad \ell = 0, \ldots, p, \tag{11.57}$$

$$R_{dx}[\ell] = \sum_{k=0}^{p} h[k]R_{xx}[k-\ell], \quad l = 0, \ldots, p, \tag{11.58}$$

which can be recognized as another form of the auto-correlation normal equations we have seen previously for AR modeling. We have

$$\begin{bmatrix} R_{xx}[0] & \ldots & R_{xx}[p] \\ \vdots & \ddots & \vdots \\ R_{xx}[p] & \ldots & R_{xx}[0] \end{bmatrix} \begin{bmatrix} h[0] \\ \vdots \\ h[p] \end{bmatrix} = \begin{bmatrix} R_{dx}[0] \\ \vdots \\ R_{dx}[p] \end{bmatrix}, \tag{11.59}$$

which has the solution,

$$\begin{bmatrix} h_{\text{MMSE}}[0] \\ \ldots \\ h_{\text{MMSE}}[p] \end{bmatrix} = \begin{bmatrix} R_{xx}[0] & \vdots & R_{xx}[p] \\ \vdots & \ddots & \vdots \\ R_{xx}[p] & \vdots & R_{xx}[0] \end{bmatrix}^{-1} \begin{bmatrix} R_{dx}[0] \\ \vdots \\ R_{dx}[p] \end{bmatrix}, \tag{11.60}$$

$$\vec{h}_{\text{MMSE}} = R^{-1}\vec{p}, \tag{11.61}$$

taking a similar form to the AR modeling case. Once again, since the matrix to be inverted is an auto-correlation matrix, and is symmetric, positive semidefinite, and Toeplitz, there exist fast, and numerically

stable algorithms for solving for the MMSE-optimal parameters [6,8]. This set of parameters minimizes the mean square error, and is hence called the MMSE-optimal set of model parameters. They are also often referred to as the Wiener solution, or the Wiener filter [8].

A deterministic formulation of the problem could be made, using the least-squares criterion, which would again lead to a set of equations in a similar form, specifically, defining the error criterion to be

$$\{h_{LS}[k]\}_{k=0}^{p} = \arg \min_{\{h[k]\}_{k=0}^{p}} \epsilon_{LS} = \sum_{n=-\infty}^{\infty} \left(d[n] - \sum_{k=0}^{p} h[k]x[n-k] \right)^2. \tag{11.62}$$

This leads to

$$\sum_{n=-\infty}^{\infty} d[n]x[n-\ell] = \sum_{k=0}^{p} h[k] \sum_{n=-\infty}^{\infty} x[n-k]x[n-\ell], \quad \ell = 0, \ldots, p, \tag{11.63}$$

$$\hat{R}_{dx}[\ell] = \sum_{k=0}^{p} h[k]\hat{R}_{xx}[k-\ell], \quad \ell = 0, \ldots, p, \tag{11.64}$$

where $\hat{R}_{dx}[\ell]$ and $\hat{R}_{xx}[\ell]$ are defined using summations over the observed data, rather than taking a statistical expectation. The resulting set of normal equations are identical, with the statistical auto- and cross-correlation functions replaced by their deterministic counterparts, arriving at the solution,

$$\begin{bmatrix} h_{LS}[0] \\ \ldots \\ h_{LS}[p] \end{bmatrix} = \begin{bmatrix} \hat{R}_{xx}[0] & \ldots & \hat{R}_{xx}[p] \\ \vdots & \ddots & \vdots \\ \hat{R}_{xx}[p] & \vdots & \hat{R}_{xx}[0] \end{bmatrix}^{-1} \begin{bmatrix} \hat{R}_{dx}[0] \\ \vdots \\ \hat{R}_{dx}[p] \end{bmatrix}, \tag{11.65}$$

$$\vec{h}_{LS} = \vec{R}^{-1}\hat{\vec{p}}. \tag{11.66}$$

1.11.3.6 Sequential parametric modeling

In certain signal processing problems, the observations may arrive sequentially and the application may require immediate output based on only the available data. In other applications, different model parameters may be suitable for different parts of the observations such that parameters are updated in time for better fit. A common approach to consolidate these is to produce the estimated parameters in a sequential manner. Unlike the previous approaches where we have the batch data $\{x[k]\}_{k=0}^{N}$, the parameters $\{x[k]\}_{k=0}^{p}$ are selected based on only currently available data $\{x[k]\}_{k=-\infty}^{n}$. As in the previous sections, a deterministic formulation of the problem could be made, using the least-squares criterion, based only on the available data as

$$\{h_{LS}[k,n]\}_{k=0}^{p} = \operatorname{argmin}_{\{h[k]\}_{k=0}^{p}} \sum_{t=-\infty}^{n} \left(d[t] - \sum_{k=0}^{p} h[k]x[t-k] \right)^2. \tag{11.67}$$

The estimated parameters are now function of time and updated as new observations arrive. To emphasize the most recent data in the estimate, the least-squares criterion is often defined over a window of observations or a weighted least squares problem is formulated, such as

$$\{h_{LS}[k,n]\}_{k=0}^{p} = \underset{\{h[k]\}_{k=0}^{p}}{\operatorname{argmin}} \sum_{t=-\infty}^{n} \lambda_{n-t} \left(d[t] - \sum_{k=0}^{p} h[k]x[t-k] \right)^2, \tag{11.68}$$

where the emphasized window length is intrinsically set with weighting parameter $0 < \lambda \leq 1$, e.g., when $\lambda = 0.9$ the most recent data is emphasized in that any samples that are more than $(\frac{1}{1-\lambda}) = 10$ samples old are weighted by $1/e$ or less, as compared with the most recent data sample. The sequential and weighted least-squares criterion leads to a similar set of normal equations

$$\begin{bmatrix} h_{LS}[0,n] \\ \dots \\ h_{LS}[p-n] \end{bmatrix} = \begin{bmatrix} \hat{R}_{xx}[0,n] & \dots & \hat{R}_{xx}[p,n] \\ \vdots & \ddots & \vdots \\ \hat{R}_{xx}[p,n] & \dots & \hat{R}_{xx}[0,n] \end{bmatrix}^2 \begin{bmatrix} \hat{R}_{dx}[0,n] \\ \vdots \\ \hat{R}_{dx}[p,n] \end{bmatrix}, \tag{11.69}$$

$$\vec{h}_{LS}[n] = \hat{R}[n]^{-1}\hat{\vec{p}}[n]. \tag{11.70}$$

However, here, the sample auto-correlation matrix and cross-correlation vector are calculated using only the available data, and hence, are time varying,

$$\hat{R}_{xx}[k-\ell,n] = \sum_{t=-\infty}^{n} \lambda^{n-t} x[t-k]x[t-\ell], \quad l = 0,\dots,p, \tag{11.71}$$

$$\hat{R}_{dx}[\ell,n] = \sum_{t=-\infty}^{n} \lambda^{n-t} d[t]x[t-\ell], \quad l = 0,\dots,p. \tag{11.72}$$

Although successful in most signal processing applications, this online processing requires calculating the sample correlation matrix, cross-correlation vector and performing inversion for each new sample, which may be computationally infeasible for some applications. However, by recognizing the time recursions in

$$\hat{R}_{xx}[k-\ell,n+1] = \lambda \hat{R}_{xx}[k-\ell,n] + x[n+1-k]x[n+1-\ell], \quad \ell = 0,\dots,p, \tag{11.73}$$

$$\hat{R}_{dx}[\ell,n+1] = \lambda \hat{R}_{dx}[\ell,n] + d[n+1]x[n+1-\ell], \quad \ell = 0,\dots,p, \tag{11.74}$$

and using the matrix inversion lemma [8], we can solve the corresponding estimation problem in a recursive form such that the estimated model parameters at time $n+1$ can be readily obtained from the estimated parameters at time n as

$$e[n] = d[n] - \hat{d}[n], \tag{11.75}$$

$$\vec{g}[n] = p[n]\vec{x}[n]\{\lambda + \vec{x}[n]^T p[n]\vec{x}[n]\}^{-1}, \tag{11.76}$$

$$p[n+1] = \lambda^{-1}p[n] - \vec{g}[n]\vec{x}[n]^T\lambda^{-1}p[n], \tag{11.77}$$

$$\vec{h}[n+1] = \vec{h}[n] + \vec{g}[n]e[n] \tag{11.78}$$

yielding the "recursive least squares" (RLS) algorithm [8], where $P[n] = \hat{R}[n]^{-1}$. While there exist more computationally efficient lattice implementations of this update without the matrix inversion lemma, the RLS algorithm is shown to be numerically more stable [8].

We observe from this recursive update that the corresponding estimated parameter vector $\vec{h}[n]$ at time n is basically updated by an additive vector, called the gain vector, times the estimation error to yield the parameters at time $n + 1$ as

$$\vec{h}[n+1] = \vec{h}[n] + \vec{g}[n]e[n], \tag{11.79}$$

where the gain vector is the inverse correlation matrix multiplied by the data vector $\vec{x}[n]$ with certain power scaling. A common approach in signal processing to reduce the computational complexity of this update is to replace the scaled inverse correlation matrix by the inverse of the trace of the correlation matrix

$$\vec{h}[n+1] = \vec{h}[n] + \frac{1}{Trace(\hat{R}[n])}\vec{x}[n]e[n]. \tag{11.80}$$

Subsequently, the inverse of the trace, which corresponds to the power in the stochastic process, is further replaced by a positive constant $\mu > 0$ as

$$\vec{h}[n+1] = \vec{h}[n] + \mu\vec{x}[n]e[n], \tag{11.81}$$

yielding the celebrated least mean squares (LMS) adaptive algorithm [9].

When the underlying stochastic model is known, i.e., the cross and auto-correlation functions are known, the parameters that minimize the MSE are evaluated by solving the normal equations

$$\vec{h}_{\text{MMSE}} = R^{-1}\vec{p}. \tag{11.82}$$

The same normal equations can be iteratively solved by applying a gradient descent algorithm [8] to the mean square error cost function, yielding a recursive update

$$\vec{h}^{(k+1)} = \vec{h}^{(k)} - \mu\nabla_{\vec{h}^{(k)}} E[e^2[n]], \tag{11.83}$$

$$\vec{h}^{(k+1)} = \vec{h}^{(k)} - \mu(\vec{p} - R\overrightarrow{h}\,(k)), \tag{11.84}$$

which converges to $\vec{h}_{\text{MMSE}}$ as k increases provided that μ is selected appropriately [8,9]. If the underlying R and p are not known, then the same iteration can be carried in time domain by replacing the expected variables by their temporal values, in other words by replacing $E[e^2[n]]$ with $e^2[n]$, to yield

$$\vec{h}^{[n+1]} = \vec{h}[n] - \mu\nabla_{\vec{h}[n]} e^2[n], \tag{11.85}$$

$$\vec{h}^{[n+1]} = \vec{h}^{[n]} - \mu(\vec{x}[n]e[n], \tag{11.86}$$

which yields the stochastic interpretation of the LMS algorithm. Hence, the LMS algorithm can be considered a gradient descent algorithm, where a "stochastic" gradient is used in the update instead of the true gradient as shown in Figure 11.8 [8].

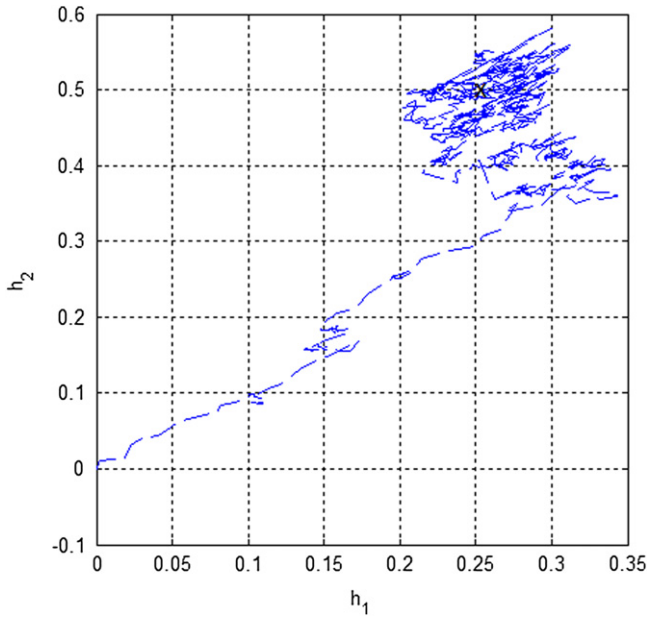

FIGURE 11.8

Graphical depiction of the convergence of the stochastic gradient (LMS) algorithm, depicted for a two tap filter initialized at $h = [0, 0]$.

1.11.3.7 Universal model order estimation

Even when a parametric model is known to be appropriate for a particular application, e.g., an AR, ARMA or MA model, we still need to decide on the number of parameters in the corresponding parametric model. Determination of the proper number of parameters to use for a parametric model is known to be a difficult problem and has a rich history in a number of applications in signal processing, machine learning and communications [8,10]. If the complete statistical properties of the underlying signals were known, then increasing the number of parameters would have only increased the modeling power, i.e., more parameters are better. However, since the model parameters should be learned from a limited amount of training data, this can cause over fitting, especially in short-data regimes, and lead to poor extrapolation outside of the training set, i.e., too many parameters can be bad. To avoid such over fitting problems, much of the early work in the model order determination literature focused on methods that are based on assumed probabilistic models, such as the minimum description length (MDL) or the Akaike information criterion (AIC) [6,11]. These methods establish a balance between modeling power of the parameters using certain maximum likelihood, or information theoretic measures, while penalizing the number of parameters using certain regularization terms. For example, for AR modeling with white Gaussian noise, the leading term of the penalty term in the MDL for a model of order p and a signal length n is given by $\frac{p}{2}\log(n)$. Hence, for a signal $d[t]$ according to the original definition of

MDL, the model order is selected by minimizing

$$\min_{p} \left\{ \min_{\{h[k]\}_{k=0}^{p}} \sum_{t=1}^{n} \left(d[t] - \sum_{k=1}^{p} h[k] d[t-k] \right)^2 + \left(\frac{p}{2} \right) \log(n) \right\}. \tag{11.87}$$

While these statistical approaches have been widely used in different signal processing applications, they require considerable statistical information on the signals of interest and can be fragile to operating in regimes in which the signals, for example, do not behave according to the assumed statistical model.

However, recent work in the machine learning and information theory communities have applied a new approach to tackling the model order selection problem [12,13]. As an example, in the universal model order selection approach, instead of making hard decisions at each instant based on an assumed statistical model, one uses a performance based mixture of all models of different orders in an attempt to perform as well as or better than the best model in the mixture. The resulting algorithms then successfully navigate the short-data record regime by placing more emphasis on lower order models, while achieving the ultimate precision of higher order models as the data record grows to accommodate them. The elegance of the universal mixture methods is that this occurs naturally by monitoring the performance of each model on the data observed so far and placing more emphasis on the results of better performing models in an online manner.

This model combination approach is successfully applied to model order selection in the context of RLS adaptive filtering when several different linear predictors, e.g., AR models, are used for prediction, regression or estimation. Clearly, the order of linear predictor heavily depends on the available amount of data and assumed statistical models. Here, instead of fixing a particular model order, one can use an explicit adaptive convex combination of different order linear predictors $\hat{x}_p[n]$, up to some order M, as

$$\hat{x}[n] = \sum_{p=1}^{M} \mu_p[n] \hat{x}_p[n], \tag{11.88}$$

where the corresponding adaptive combination weights $\mu_p[n]$ are calculatedly intuitively as $\mu_p[n] \sim \exp\left(-\sum_{t=1}^{n} (x(t) - \hat{x}_p[t])\right)$, i.e., based on the performance of th predictor on the observed data so far. An example weighted mixture that combines m adaptive filters is shown in Figure 11.9. The output is simply calculated as a performance-weighted mixture. For these applications, an efficient lattice filter structure can be used to not only implement each of the elements of the competing class, but to also construct the universal predictor output all with complexity that is only linear in the largest model order in the competing class of models [9]. Such a model combination is shown to achieve, for any n [12]

$$\frac{1}{n} \sum_{n=1}^{N} (x[n] - \hat{x} - \hat{x}[n])^2 \leq \min_{p<M} \frac{1}{n} \sum_{n=1}^{N} (x[n] - \hat{x}_p[n])^2 + \frac{1}{n} \ln(M). \tag{11.89}$$

The ability to asymptotically achieve the performance of the best model order in the set of models gives rise to the term "universal filter" and hence such methods are often called "universal" methods.

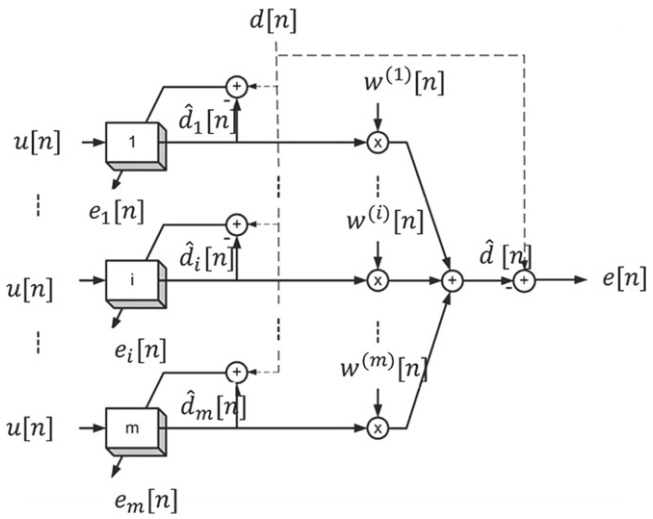

FIGURE 11.9

Adaptively combining the outputs of multiple adaptive filters.

1.11.3.8 Universal linear predictor

In parameter estimation, the universal algorithms approach can be generalized to construct estimators that are, in a deterministic sense, min-max optimal [13]. As an example, once again let us focus on AR parameter estimation under the deterministic total square estimation error. However, instead of constructing sequential normal equations, as in the previous section, we hypothetically implement all of the continuum of AR models for each $\vec{h} \in R^p$ and contruct a sequential estimate based on a performance-weighted mixture of all these AR models as in the previous section. By taking an implicit performance-weighted mixture over the outputs of all adaptive filters $\vec{h} \in R^p$ and using a priori Gaussian weighting $p(\hat{h})$, the integral

$$\int_{R^p} \vec{h}^T \vec{x}[n] p(\vec{h}) \sum_{t=1}^{n-1} e^2[t] d\vec{h}, \tag{11.90}$$

analogous to the weighted combination in the previous section, can be evaluated in closed form and leads to a diagonally loaded RLS adaptive filtering algorithm [12]

$$\vec{h}_{uni}[n] = \vec{R}[n-1]^{-1} \hat{\vec{p}}[n]. \tag{11.91}$$

However, in this new recursive update, we have $\hat{R}[n-1] = \sum_{t=1}^{n-1} \vec{x}[t]\vec{t}^T + \delta I$. Hence, the diagonal loading term is the missing ingredient that takes the standard RLS adaptive filtering algorithm from being simply universal to attaining the min-max optimal performance [13]. When applied to any bounded and arbitrary sequence, this universal recursive algorithm yields the min-max optimal performance bound

for any N as

$$\frac{1}{n}\sum_{n=1}^{N}\left(x[n]-\vec{h}_{uni}[n]^T\vec{x}[n]\right)^2 \leq \min_{\vec{h}}\frac{1}{n}\sum_{n=1}^{N}\left(d[n]-\vec{h}^T\vec{x}[n]\right)^2+\frac{1}{n}0(p\ln N). \tag{11.92}$$

This is min-max optimal, in that no other approach can achieve a lower "regret" than the $\frac{1}{n}0(p\ln N)$ term above [14].

1.11.3.9 Universality with respect to classes that vary in space: nonlinear classes/piecewise in space

Linear AR, MA, or ARMA models discussed in the previous sections are extensively used in signal processing due to their tractability and adequate accuracy in modeling [9]. Recently, nonlinear models and those based on piecewise linear and locally linear approximations have gained significant attention as they capture the salient characteristics of many physical phenomena. Nonlinear parametric estimation methods, for example, Volterra filters [8] have proven attractive for a number of applications. By removing structural constraints on linearity, nonlinear models are perhaps more appropriate for a variety of data exhibiting saturation effects, threshold phenomena or other nonlinear behavior.

Piecewise linear modeling can be viewed as a natural extension to linear modeling, in which the space spanned by past observations is partitioned into a union of disjoint regions over each of which an affine model is fitted. In each region, a linear model can be estimated, and as the number of regions grows, the piecewise linear model can better approximate any smoothly varying nonlinear estimator. As an example to a piecewise linear estimator, suppose the space of past observations $\vec{x}[n]\in R^p$ is partitioned into k disjoint regions $R^p=U_{k=1}^{K}V_k$. Based on this partitioning, the observations can be divided into k disjoint sets, where k different normal equations are written as

$$\vec{R}_k=\sum_{n=1}^{N}\vec{x}[n]\vec{x}[n]^T I_k[n]), \tag{11.93}$$

$$\vec{p}_k=\sum_{n=1}^{N}d[n]\vec{x}[n]^T I_k(\vec{x}[n]), \tag{11.94}$$

and $I_k(\vec{x}[n])$ is the indicator function of the kth region, i.e., $I_k(\vec{x}[n])=1$ if $\vec{x}[n]\in V_k$, for each region. This partitioning of the observations yields a linear estimator,

$$\vec{h}_k=\hat{R}_k^{-1}\hat{\vec{p}}_k, \tag{11.95}$$

$k=1,\ldots,k$, at each region, yielding a piecewise linear model. A similar formulation is possible when the data arrives sequentially, however, in that case, the AR parameters are estimated sequentially using time dependent normal equations. As an example, for sequential processing, we can run k different LMS algorithms, one for each region, to get an adaptive piecewise linear model trained as

$$\vec{h}_k[n+1]=\vec{h}_k[n]+\mu e_k[n]\vec{x}[n]I_k(\vec{x}[n]), \tag{11.96}$$

where $e_k[n] = d[n] - \vec{h}_k[n]^T \vec{x}[n]$. These types of piecewise linear models have been referred to in the signal processing literature as "nonlinear autoregressive models" and in the signal processing and statistics literature as "self-exciting threshold auto regressive models," and have been used in modeling a wide range of data in fields ranging from population biology to econometrics to glottal flow in voiced speech [15].

To fully exploit the potential modeling power of piecewise linear estimators, we need to be cautious while selecting the number of regions and boundaries of these regions. Piecewise linear models can approximate a wide class of nonlinear models, which satisfy certain regularity conditions. As the number of regions increases, we point out that the normal equations are calculated using a finite number of observations assigned to each region, which on a per-region basis, necessarily decreases with the number of regions. Hence, over fitting may occur if there are not enough observations assigned to a particular region. Clearly, how the boundaries are selected also affects the effectiveness of the piecewise linear models and how the observations are assigned to particular regions. These design considerations are especially severe in sequential processing, where the observations can only accumulate in time and available data is sparse in the initial phase of the adaptation.

One solution to avoid possible over fitting is to include the boundaries of regions as design parameters along with the corresponding AR parameters. Along these lines, recently, context tree based methods have been used to compactly represent a doubly exponential number of partitions of the space of past observations and provide an elegant way to design the partition of the past observations together with the AR parameters.

An example context tree is given in Figure 11.10 which partitions the space of past observations, where the depth of this context tree is 2, i.e., $K = 2$ [16]. For a context tree of depth $K = 2$, we have four leaves, where each leaf is assigned to a region of the space of past observations. Subsequently, each node on the context tree is assigned to a region that is the union of regions assigned to its children nodes. Using this context tree, we can define different partitions of the space of past observations as the union of regions assigned to nodes and the leaves. A context tree of depth can define a doubly exponential number of different partitions, e.g., in Figure 11.10, we present five different partitions of the space constructed by the context of depth $K = 2$. We represent a partition of the space of past observations as $P_k = \{V_{k,1}, \ldots, V_{k,K_k}\}$, where $\mathbf{U}_{j=1}^{K_k} V_{k,j}$ and each $V_{k,j}$ assigned to a node on the context tree. Each of these partitions can be used to construct a piecewise linear model since they define complete partitions of the space of past observations. Some of the partitions are coarser and require fewer data samples to accurately calculate their linear estimators, while, as an example, the partition corresponding to the leaves has the finest partitioning and requires substantially more data for training. However, if there is sufficient data, then the finest partition naturally achieves the best modeling accuracy.

By using the context tree weighting method, we can construct a sequential piecewise AR model that hypothetically constructs all the piecewise linear estimators corresponding to all of these different partitions. This context tree based estimator achieves the performance of the best piecewise region with the corresponding sequential estimated parameters in each node as

$$\sum_{n=1}^{N} (x[n] - \vec{h}_{ctw}^T[n]\vec{x}[n])^2 \leq \min_k \left\{ \sum_{n=1}^{N} (x[n] - \vec{h}_{P_k}^T[t]\vec{x}[n])^2 + C(P_k) \right\}, \tag{11.97}$$

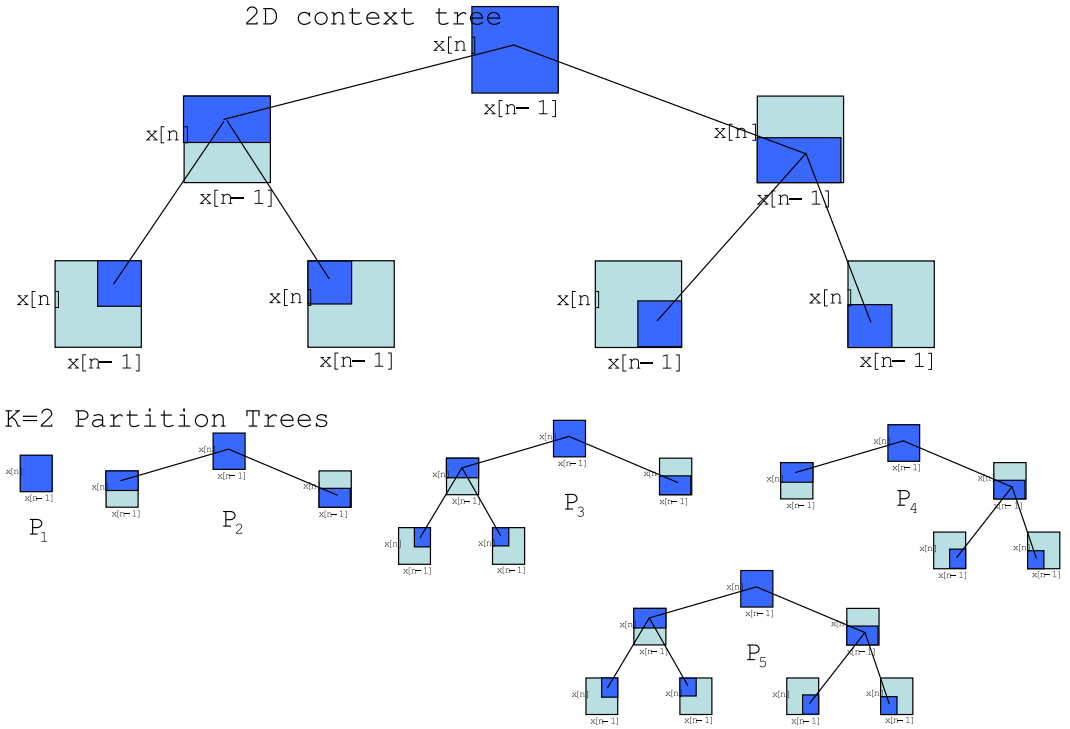

FIGURE 11.10

A context tree of depth $K = 2$, which has $K = 4$ leaves and 3 inner nodes. This tree can represent five different partitions, $P_1, \ldots, P_5$, where each partition defines a different piecewise linear estimator.

where $C(P_k)$ is a certain constant that depends on the partition P_k, $\vec{h}_{P_k}[n]$ are the sequentially estimated AR parameters by the piecewise estimator assigned to the P_k partition. As an example, if the LMS algorithms are used in each node to train the corresponding linear models in different regions, then we have $\vec{h}_{P_k}[n] = \vec{h}_{V_{k,j}}[n]$ if $\vec{x}[n] \in V_{k,j}$, and

$$\vec{h}_{V_{k,j}}[n+1] = \vec{h}_{V_{k,j}}[n] + \mu e_{V_{k,j}}[n]\vec{x}[n] I_{V_{k,j}}(\vec{x}[n]). \tag{11.98}$$

We can use different adaptation methods, e.g., the RLS algorithm instead of the LMS algorithm, in each node or region such that only the update equation should be changed accordingly. This implies that the context tree algorithm can uniformly achieve the performance of the best partition that can be represented on the context tree for any bounded deterministic signal, where the computational complexity of this algorithm is only in the order of the depth of the context tree.

1.11.3.10 Universality with respect to classes that vary in time

For non-stationary data models or time varying environments, the best choice of an adaptive estimator as well as the structure of this best estimator may change over time. As an example, different order AR models are needed to accurately model voiced and unvoiced time segments of a speech signal [3]. While RLS algorithm is known to converge faster than the LMS algorithm, since it minimizes a true least square error criterion, the LMS algorithm is known to better track certain non-stationary data [8]. In classical adaptive filtering, this kind of time variation is usually overcome by incorporating windowing or weighting by defining appropriate cost functions as done in the previous sections.

Recently, the universal model combination approaches gave rise to a host of model combination methods, where several different sequential estimation methods are combined in order to outperform the best in the combination [12]. In this sense, instead of using windowing or weighting, the best of choice of algorithms are intrinsically selected by the underlying data based on the performance so far. For example, by using a combination of adaptive algorithms that are fast tracking along with others that have superior steady state mean square error performance, one can strike a balance and achieve the tracking performance of the fast-converging adaptive filter, while maintaining the superior steady state performance of the slower, more precise adaptive filter counterpart.

Hence, rather than pushing the time variation into a known statistical model or adaptation, one can try to exploit the time varying nature of the best choice of algorithm for any given realization. As an example, given the batch data we can partition the observations in time into contiguous segments

$$\{\vec{x}[1],\ldots,\vec{x}[n_1-1]\}\{\vec{x}[n_1],\ldots,\vec{x}[n_2-1]\}\ldots\{\vec{x}[n_k],\ldots,\vec{x}[N]\} \tag{11.99}$$

and construct independent normal equations using

$$\widehat{R}_k = \sum_{n=n_{k-1}+1}^{n_k} \vec{x}[n]\vec{x}[n]^T, \tag{11.100}$$

$$\hat{\vec{p}}_k = \sum_{n=n_{k-1}+1}^{n_k} x[n]\vec{x}[n], \tag{11.101}$$

where $n_0 = 0$. This yields K independent AR models one for each contiguous segment as

$$\vec{h}_k = \hat{R}_k^{-1}\hat{\vec{p}}_k, \tag{11.102}$$

$$\vec{h}[n] = \vec{h}_k, \quad n_{k-1}+1 \le n \le n_k. \tag{11.103}$$

Since each normal equation only depends on the observations belonging to a specific segment, this adaptive processing can readily track variations in time. However, the best partition, i.e., how to optimally divide the observations in time can only be chosen after observing the whole data. Hence, we cannot construct a sequential version of such optimization and are forced to resort to windowing or weighting to emulate such piecewise partition in time.

However, inspired by the universal source coding methods, rather than trying to find the best partition (possible best switching points) or the best number of transitions, we can construct a sequential adaptive algorithm that simply achieves the performance of the best partition directly and simultaneously for all

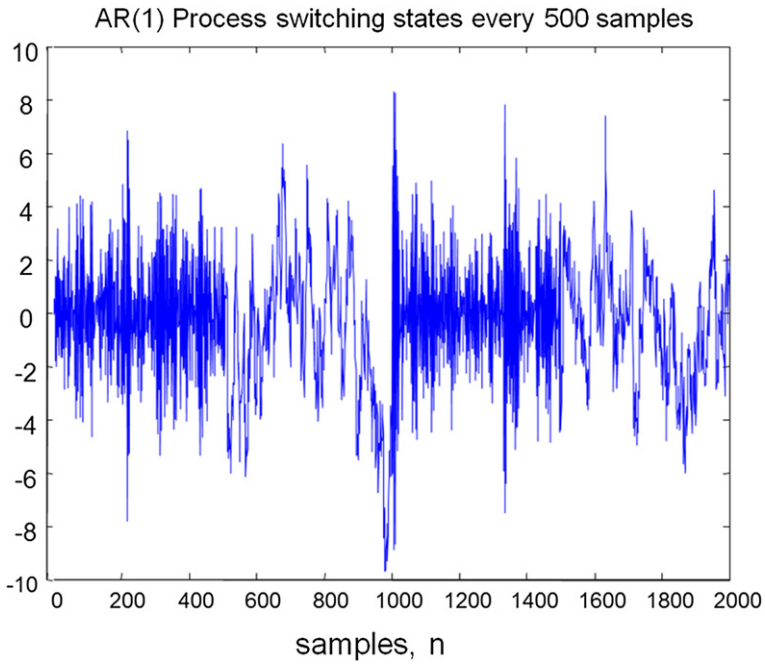

FIGURE 11.11

A time-varying AR(1) model used as input to the predictor that permits switching among different linear predictors in time.

different partitions and for any possible observation sequence. This switching approach has been successfully applied by a number of researchers in a variety of problems, including tracking the best among a fixed class of algorithms as well as tracking the best from parametrically-continuous classes of algorithms [13]. By using transition diagrams, we can construct sequential algorithms that asymptotically achieve the performance of the best switching algorithm, i.e.,

$$\sum_{n=1}^{N}(x[n]-\vec{h}_{ctw}^{T}[n]\vec{x}[n])^2 \leq \min_{n_1,\ldots,n_K,\vec{h}_1,\ldots,\vec{h}_K}\left\{\sum_{k=0}^{K}\sum_{n=n_k+1}^{n_k}(x[n]-\vec{h}_k^{T}\vec{x}[n])^2+L(K)\right\} \quad (11.104)$$

tuned to the underlying sequence of observations without any stochastic assumptions, where $L(K)$ is a constant that only depends on the number of contiguous segments.

The performance of a universal linear predictor that switches in time is demonstrated with the following example.

In this example, we are tracking a process similar to a speech sample such that the state of the process switches states abruptly every 500 samples, as shown in Figure 11.11. The state of the process at each segment is represented by an AR model driven by a Gaussian noise [8]. Figure 11.12 shows the mean

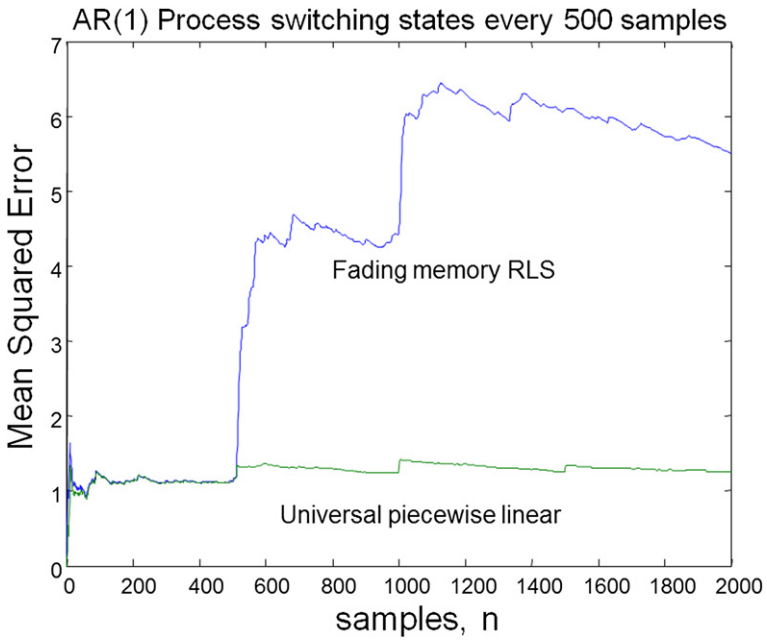

FIGURE 11.12

Mean square prediction error exhibited by (i) a fading memory RLS linear predictor and (ii) the universal piecewise constant (in time) linear predictor.

square error that results when we attempt to track the process using a fading memory RLS algorithm versus that obtained using a universal piecewise linear algorithm.

It can be seen that the universal piecewise linear algorithm recovers from the state transitions much more quickly, since it competes against the best switching pattern, while the fading memory RLS algorithm cannot recover in the short-data regime between transitions. Therefore, in such a situation, a universal piecewise linear model gives superior performance relative to the standard single filter model.

Relevant Theory: Signal Processing Theory, Machine Learning and Statistical Signal Processing

See this Volume, Chapter 2 Continuous-Time Signals and Systems
See this Volume, Chapter 3 Discrete-Time Signals and Systems
See this Volume, Chapter 4 Random Signals and Stochastic Processes
See this Volume, Chapter 6 Digital Filter Structures and their Implementation
See this Volume, Chapter 9 Discrete Multi-Scale Transforms in Signal Processing
See this Volume, Chapter 11 Parametric Estimation
See this Volume, Chapter 12 Adaptive Filters
See this Volume, Chapter 14 Learning Theory
See this Volume, Chapter 17 Online Learning

See this Volume, Chapter 25 A Tutorial on Model Selection
See Vol. 3, Chapter 2 Model Order Selection
See Vol. 3, Chapter 3 Non-Stationary Signal Analysis Time-Frequency Approach
See Vol. 3, Chapter 8 Performance Analysis and Bounds

References

[1] A. Oppenheim, R.W. Schafer, J.R. Buck, Discrete-Time Signal Processing, second ed., Prentice Hall, 1999.
[2] L. Rabiner, B. Gold, Theory and Application of Digital Signal Processing, Prentice-Hall, Inc., Englewood Cliffs, 1975.
[3] L.R. Rabiner, R.W. Schafer, Introduction to Digital Speech Processing, Prentice Hall, Inc., 1978.
[4] W. Feller, An Introduction to Probability Theory and its Applications, Wiley, 1968.
[5] B. Hajek, An Exploration of Random Processes for Engineers, 2011. Retrieved from An Exploration of Random Processes for Engineers: <http://www.ifp.illinois.edu/∼hajek/Papers/randomprocesses.html>.
[6] S. Kay, Fundamentals of Statistical Signal Processing, Estimation Theory, vol. 1, Prentice Hall, 1993.
[7] J. Mahkoul, Linear prediction: a tutorial review, Proc. IEEE (1975) 561–580.
[8] A.H. Sayed, Fundamentals of Adaptive Filtering, Wiley-IEEE Press, 2003.
[9] M.L. Honig, D.G. Messerschmitt, Adaptive Filters: Structures, Algorithms and Applications, Springer, 1984.
[10] J.G. Proakis, Digital Communications, McGraw-Hill, 1983.
[11] H.V. Poor, An Introduction to Signal Detection and Estimation, Springer, New York, 1988.
[12] A.C. Singer, M. Feder, Universal linear prediction by model order weighting, IEEE Trans. Signal Process. 47 (10) (1999) 2685–2699.
[13] N. Cesa-Bianchi, G. Lugosi, Prediction Learning and Games, Cambridge University Press, 2006.
[14] A.C. Singer, S.S. Kozat, M. Feder, Universal linear least squares prediction upper and lower bounds, IEEE Trans. Inform. Theory 48 (8) (2002) 2354–2362.
[15] O.J. Michel, A.O. Hero, A.E. Badel, Tree-structured nonlinear signal modeling and prediction, IEEE Trans. Signal Process. 47 (11) (1999) 3027–3040.
[16] S.S. Kozat, A.C. Singer, G.C. Zeitler, Universal piecewise linear prediction via context trees, IEEE Trans. Signal Process. 55 (7) (2007) 3730–3745.