ELEC60002 Statistical Signal Processing and Inference

Timothy Chung

Spring 2024

Contents

Chapter 0

Foreword

The notes are intended to be a gentle introductory supplement to the well-documented course notes, and not a full replacement with all the bells and whistles (e.g. real life use-cases, mathematical examples). The full notes are available on Professor Danilo Mandic's website here.

Thus, a better title for this document would be 'Elements of Statistical Signal Processing and Inference'.

Chapter 1

Introduction

1.1 Applications of Statistical Signal Processing

Uses of Statistical Signal Processing

- Forecasting financial data
- Supply-demand modelling
- Pandemic spread modelling

Main theme: making sense from data based on historical observations.

There are two kinds of statistics: descriptive statistics used to describe data, and inferential statistics that try to gather generalisations from random observations.
Uses of Statistical Inference

- Adaptive learning algorithms like noise-cancelling headphones
- Neural networks for classification, prediction, denoising
- Financial modelling

Prerequisites:

- Basic statistics knowledge, refer to slide 5 of Lecture 1 notes.
- Signals and systems knowledge from 2nd year.

Learning goals:

- Random signals (real-world discrete-time), their properties and statistical descriptors
- Linear stochastic models
- Grounding in linear estimation theory
- Parametric (model-based) and nonparametric (data-driven) modelling
- Optimal estimators for random signals with performance bounds
- Least squares methods
- Adaptive estimation for nonstationary data

Chapter 2

Linear Stochastic Models

2.1 Objectives

- Introduce linear stochastic models for real world data
- Understand stochastic processes for noise
- ARMA models, partial correlations, and optimal model order selection criteria (MDL, AIC,...)

Wold's Decomposition Theorem **Theorem 2.1.1**

Any covariance-stationary time series can be decomposed into two different parts: **deterministic** and **stochastic**.

Let w denote a white process, $x_r[n]$ a regular random process, and $x_p[n]$ a predictable process such that $x_r[n] \perp x_p[n]$.

$$x[n] = x_p[n] + x_r[n] \tag{2.1}$$

$$= x_p[n] + \sum_{j=1}^{q} b_j w[n-j] \tag{2.2}$$

The condition requires $\mathrm{E}[x_r[m]x_p[n]] = 0$.

Deterministic vs Stochastic **Definition 2.1.1**

- Deterministic means a component can be precisely described by an equation without any randomness, following a predictable pattern that can be determined exactly for any point in time, e.g. a sine wave.
- Stochastic means a component described by a random process. It cannot be described by a simple equation but can be represented using a probability distribution, such as WGN (Filtered White Gaussian) noise.

2.2 WSS Process

A WSS process has two conditions:

1. The mean of the process is constant for all time, i.e., $\mathrm{E}[X_t] = \mu$ for all t, where μ is a constant.
2. The autocovariance function $R_x(t, t+\tau) = \mathrm{Cov}(X_t, X_{t+\tau})$ depends only on the time lag τ and not on time t itself, i.e., $R_x(t, t+\tau) = R_x(\tau)$.

The general form for the power spectrum of a WSS process is

$$P_x(e^{j\omega}) = \sum_{k=1}^{N} \alpha_k \delta(\omega - \omega_k) + P_{x_r}(e^{j\omega}) \tag{2.3}$$

Linearly Deterministic Process **Definition 2.2.1**

A covariance-stationary process is called linearly deterministic if $p(x[n] \mid x[n-1], x[n-2], \ldots) = x[n]$. It can be predicted correctly with zero error if we know its entire past data $x[n-1], x[n-2], \ldots$

2.3 ARMA Models

ARMA models are a linear stochastic model.

Autoregressive (AR) filters use an all-pole system and Moving Average (MA) filters use an all-zero system. An Autoregressive Moving Average (ARMA) filter utilises both poles and zeroes.

In ARMA modelling we filter white noise $w[n]$ with a causal linear shift-invariant filter (transfer function $H[z]$ that has p poles and q zeroes.

$$X(z) = H(z)W(z) \quad \Rightarrow \quad H(z) = \frac{B_q(z)}{A_p(z)} = \frac{\sum_{k=0}^{q} b_k z^{-k}}{1 + \sum_{k=1}^{p} a_k z^{-k}} \tag{2.4}$$

If our filter is WSS, then $x[n]$ is WSS as well. To show this, multiply both sides of Equation 2.4 by $x[n-k]$ can calulating expectation, we have

$$r_{xx}(k) = \underbrace{\sum_{l=1}^{p} a_l r_{xx}(k-l)}_{\text{easy to calculate}} + \underbrace{\sum_{l=0}^{q} b_l r_{xw}(k-l)}_{\text{can be complicated}} \tag{2.5}$$

Note that since $X(z) = H(z)W(z)$ the random processes $x[n]$ and $w[n]$ are related by a linear difference equation with constant coefficients. This is

$$\begin{aligned} \text{ARMA(p, q)} = H(z) = &\frac{B(z)}{A(z)} = \frac{\sum_{k=0}^{q} b_k z^{-k}}{1 + \sum_{k=1}^{p} a_k z^{-k}} \\ x[n] = &\underbrace{\sum_{l=1}^{p} a_l x[n-l]}_{\text{autoregressive}} + \underbrace{\sum_{l=0}^{q} b_l w[n-l]}_{\text{moving average}} \end{aligned} \tag{2.6}$$

Note that for $H(z)$, coefficients must be absolutely summable. For the process to be stationary, $\sum_{j=0}^{\infty} |b_j| < \infty$, and for it to be invertible, $\sum_{j=0}^{\infty} |a_j| < \infty$.

2.4 AR Processes

An autoregressive process of order p, denoted by $AR(p)$ can be described by:

$$x[n] = a_1 x[n-1] + a_2 x[n-2] + \ldots + a_p [n-p] + w[n] \tag{2.7}$$

$$= \sum_{i=1}^{p} a_i x[n-i] + w[n] \tag{2.8}$$

$$= \mathbf{a}^T \mathbf{x}[n] + w[n] \tag{2.9}$$

2.4.1 ACF of an AR Process

We start by finding $x[n-k]x[n]$:

$$x[n-k]x[n] = a_1 x[n-k]x[n-1] + a_2 x[n-k]x[n-2] + \cdots + a_p x[n-k]x[n-p] + x[n-k]w[n] \tag{2.10}$$

When $k > 0$, $\mathrm{E}\{x[n-k]w[n]\} = 0$ due to both processes being orthogonal to each other. We are left with:

$$r_{xx}(k) = \begin{cases} a_1 r_{xx}(1) + a_2 r_{xx}(2) + \cdots + a_p r_{xx}(p) + \sigma_w^2, & \text{for } k = 0 \\ a_1 r_{xx}(k-1) + a_2 r_{xx}(k-2) + \cdots + a_p r_{xx}(k-p), & \text{for } k > 0 \end{cases} \tag{2.11}$$

2.4.2 Normalised ACF of an AR Process

We can normalise by dividing by $r_{xx}(0)$ to get $\rho(k) = r_{xx}(k)/r_{xx}(0)$.

$$\rho(k) = a_1\rho(k-1) + a_2\rho(k-2) + \cdots + a_p\rho(k-p) \quad k > 0 \tag{2.12}$$

2.4.3 Variance of an AR Process

For $k = 0$, the $\mathrm{E}\{x[n-k]w[n]\}$ term contributes σ_w^2 to variance, and

$$r_{xx}(0) = a_1 r_{xx}(-1) + a_2 r_{xx}(-2) + \cdots + a_p r_{xx}(-p) + \sigma_w^2 \tag{2.13}$$

Dividing by $r_{xx}(0) = \sigma_x^2$, we get:

$$\sigma_x^2 = \frac{\sigma_w^2}{1 - \rho_1 a_1 - \rho_2 a_2 - \cdots - \rho_p a_p} \tag{2.14}$$

2.4.4 Power Spectrum of an AR Process

Recall the formula for otuput power of a linear system, $P_{xx} = |H(z)|^2 P_{ww} = H(z)H^*(z)P_{ww}$. We then have

$$P_{xx}(f) = \frac{2\sigma_w^2}{\left|1 - a_1 e^{-j2\pi f} - \cdots - a_p e^{-j2\pi pf}\right|^2} \quad 0 \le f \le 1/2 \tag{2.15}$$

2.5 MA Processes

A moving average process of order q, $MA(q)$, is given by:

$$\begin{aligned} x[n] =& w[n] + b_1 w[n-1] + \cdots + b_q w[n-q] && (2.16)\\ =& w[n] + \sum_{i=1}^{q} b_i w[n-i] && (2.17)\\ =& \mathbf{b^T w}[n] + w[n] && (2.18) \end{aligned}$$

2.5.1 ACF of an MA process

Note that the ACF has a cutoff after lag q.

$$r_{xx}(k) = \mathrm{E}\big[(w[n] + b_1 w[n-1] + \cdots + b_q w[n-q])(w[n-k])\big] \tag{2.19}$$

2.5.2 Variance of an MA process

We sub in $k = 0$ into the ACF to obtain the variance:

$$r_{xx}(0) = (1 + b_1^2 + \cdots + b_q^2)\sigma_w^2 \tag{2.20}$$

2.5.3 Power Spectrum of an MA process

Since a moving average filter has a transfer function of all zeroes, and no poles except at the origin.which is known as an ARMA process.

$$P(f) = 2\sigma_w^2 \left|1 + b_1 e^{-j2\pi f} + b_2 e^{-j4\pi f} + \cdots + b_q e^{-j2\pi qf}\right|^2 \tag{2.21}$$

An MA process has a limited ability to accurately represent time series with spectra that have sharp peaks (high power at specific frequencies). This is because the MA model, being a sum of weighted noise components, tends to produce a smoother spectrum without sharp features. To model time series with sharp spectral peaks (like those that might be seen in signals dominated by sinusoidal components), one would typically need AR or ARMA processes.

2.6 Duality of AR and MA processes

Because of duality between IIR and FIR (infinite and finite impulse response) filters, every AR process has an MA representation. Take for example AR(1):

$$x[n] = a_1 x[n-1] + w[n] \quad \Leftrightarrow \quad \sum_{j=0}^{\infty} b_j w[n-j] \tag{2.22}$$

2.7 Yule-Walker and the ACF: Motivations

Say we want to find the coefficients for an AR(1) process:

$$x[n] = a_1 x[n-1] + w[n] \tag{2.23}$$

The case of $p = 1$ is trivial. We form the over-determined system

$$\underbrace{\begin{pmatrix} x_2 \\ x_3 \\ \vdots \\ x_N \end{pmatrix}}_{\mathbf{b}} = \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{pmatrix}}_{\mathbf{A}} a_1 \tag{2.24}$$

And solve with the least-squares estimator

$$\hat{a_1} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b} = \frac{\sum_{i=1}^{N-1} x_i x_{i+1}}{\sum_{i=1}^{N-1} x_i^2} = \frac{c_1}{c_o} = r_1 \tag{2.25}$$

where c_i, r_i refer to the i-th autocovariance and correlation coefficients respectively.

For the case of $p = 2$, with process

$$x[n] = a_1 x[n-1] + a_2 x[n-2] + w[n] \tag{2.26}$$

We have

$$\underbrace{\begin{pmatrix} x_3 \\ x_4 \\ \vdots \\ x_N \end{pmatrix}}_{\mathbf{b}} = \underbrace{\begin{pmatrix} x_2 & x_1 \\ x_3 & x_2 \\ \vdots & \vdots \\ x_{N-1} & x_{N-2} \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} a_1 \\ a_2 \end{pmatrix}}_{\mathbf{a}} \tag{2.27}$$

But notice how that as the order grows, it becomes more computationally intensive to compute the inverse in $\hat{\mathbf{a}} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$ as $\mathbf{A^T A}$ is not guaranteed to be diagonal, so inverting it becomes difficult.

Instead, we can derive a more efficient process for any order p. From Equation 2.12, we note that

$$\rho(k) = a_1\rho(k-1) + a_2\rho(k-2) + \cdots + a_p\rho(k-p) \quad k > 0$$

We can list the equations as such, notating $\rho(k)$ as ρ_k:

$$\begin{aligned}
\rho_1 &= a_1\rho_0 + a_2\rho_1 + a_3\rho_2 + \cdots + a_{p-1}\rho_{p-2} + a_p\rho_{p-1} \\
\rho_2 &= a_1\rho_1 + a_2\rho_0 + a_3\rho_1 + \cdots + a_{p-1}\rho_{p-3} + a_p\rho_{p-2} \\
\rho_{p-1} &= a_1\rho_{p-2} + a_2\rho_{p-3} + a_3\rho_{p-4} + \cdots + a_{p-1}\rho_0 + a_p\rho_1 \\
\rho_p &= a_1\rho_{p-1} + a_2\rho_{p-2} + a_3\rho_{p-3} + \cdots + a_{p-1}\rho_1 + a_p\rho_0
\end{aligned}$$

Since $\rho_0 = 1$, we have:

$$\underbrace{\begin{pmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \rho_{p-1} \\ \rho_p \end{pmatrix}}_{\mathbf{r}} = \underbrace{\begin{pmatrix} 1 & \rho_1 & \rho_2 & \cdots & \rho_{p-2} & \rho_{p-1} \\ \rho_1 & 1 & \rho_1 & \cdots & \rho_{p-3} & \rho_{p-2} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ \rho_{p-2} & \rho_{p-3} & \rho_{p-4} & \cdots & 1 & \rho_1 \\ \rho_{p-1} & \rho_{p-2} & \rho_{p-3} & \cdots & \rho_1 & 1 \end{pmatrix}}_{\mathbf{R}} \underbrace{\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_{p-1} \\ a_p \end{pmatrix}}_{\mathbf{a}} \tag{2.28}$$

This is the form $\mathbf{Ra} = \mathbf{r}$, where $\mathbf{R}$ is a square coefficient matrix that is full rank and symmetric, and invertability is guaranteed. We can then go on to calculate $\hat{\mathbf{a}} = \mathbf{R}^{-1}\mathbf{r}$.

2.8 Yule-Walker and the PACF: Motivations

The PACF stands for the Partial Autocorrelation function, which shows the relationship between an observation in a time series with observations at prior time steps, with the relationships of intervening observations removed.

It is a vector π defined by

$$\pi(k) = \begin{cases} 1 & \text{if } k = 0 \\ a_{kk} & \text{if } k \geq 1 \end{cases} \tag{2.29}$$

where a_{kk} is the last component of $\mathbf{a}_k = [a_{k1}, a_{k2}, \ldots, a_{kk}]^T$ from $\mathbf{a}_k = \mathbf{R}_k^{-1}\mathbf{r}_k$. We denote a_{kj} to be the jth coefficient in an autoregressive representation of order k, at Equation 2.28.

$\rho_k = a_1\rho_{k-1} + a_2\rho_{k-2} + \cdots + a_p\rho_{k-p} \quad k > 0$ now becomes:

$$\rho_j = a_{kj}\rho_{j-1} + \cdots + a_{k(k-1)}\rho_{j-k+1} + a_{kk}\rho_{j-k} \quad j = 1, 2, \ldots, k \tag{2.30}$$

$$\underbrace{\begin{pmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \rho_{k-1} \\ \rho_k \end{pmatrix}}_{\mathbf{r}} = \underbrace{\begin{pmatrix} 1 & \rho_1 & \rho_2 & \cdots & \rho_{k-2} & \rho_{k-1} \\ \rho_1 & 1 & \rho_1 & \cdots & \rho_{k-3} & \rho_{k-2} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ \rho_{k-2} & \rho_{k-3} & \rho_{k-4} & \cdots & 1 & \rho_1 \\ \rho_{k-1} & \rho_{k-2} & \rho_{k-3} & \cdots & \rho_1 & 1 \end{pmatrix}}_{\mathbf{R}} \underbrace{\begin{pmatrix} a_{k1} \\ a_{k2} \\ \vdots \\ a_{k(k-1)} \\ a_{kk} \end{pmatrix}}_{\mathbf{a}} \tag{2.31}$$

We can solve for $k = 1, 2 \ldots$ manually:

$$a_{11} = \rho_1, \quad a_{22} = \frac{\rho_2 - \rho_1^2}{1 - \rho_1^2}, \quad a_{33} = \frac{\begin{vmatrix} 1 & \rho_1 & \rho_1 \\ \rho_1 & 1 & \rho_2 \\ \rho_2 & \rho_1 & \rho_3 \end{vmatrix}}{\begin{vmatrix} 1 & \rho_1 & \rho_2 \\ \rho_1 & 1 & \rho_1 \\ \rho_2 & \rho_1 & 1 \end{vmatrix}}, \quad \text{etc} \tag{2.32}$$

The partial autocorrelation function at lag k, denoted $\pi(k)$ (or equally the AR coefficient a_{kk}), measures the linear relationship between $x(n)$ and $x(n-k)$, once we have removed the influence of $x_{n-1}, \ldots, x_{n-k+1}$, i.e.,

$$a_{kk} = \text{corr}\left(x(n) - \hat{x}(n), x(n-k) - \hat{x}(n-k)\right).$$

The PACF is used to determine the order p of an AR Model (when we are trying to model the data with an AR model), as when lag k reaches p, PACF should drop off, indicating no linear relationship beyond that point.

Therefore, for an AR(p) process, the PAC a_{kk} is nonzero for all $k \leq p$ and zero everywhere else. In practice, it is difficult to guarantee this for real world data, so a small threshold for tolerance is needed.

2.9 AR and MA in Depth

2.9.1 Variance and Power Spectrum of an AR(1) process

Variance:

$$\sigma_x^2 = \frac{\sigma_w^2}{1 - \rho_1 a_1} = \frac{\sigma_w^2}{1 - a_1^2} \tag{2.33}$$

Power Spectrum: Note how the flat PSD of WGN is shaped according to the position of the pole of AR(1) model, determining if a filter is a high pass filter or a low pass filter.

$$P_{xx}(f) = \frac{2\sigma_w^2}{|1 - a_1 e^{-j2\pi f}|^2} = \frac{2\sigma_w^2}{1 + a_1^2 - 2a_1 cos(2\pi f)} \tag{2.34}$$

2.10 Examples of Modelling with ARMA

See the slides, page 42 onwards.

2.11 Summary: AR and MA Processes

1. A stationary finite AR(p) process can be represented as an infinite order MA process. A finite MA process can be represented as an infinite AR process.
2. The finite MA(q) process has an Autocorrelation Function (ACF) that is zero beyond q. For an AR process, the ACF is infinite in length and consists of a mixture of damped exponentials and/or damped sine waves.
3. Finite MA processes are always stable, and there is no requirement on the coefficients of MA processes for stationarity. However, for invertibility, the roots of the characteristic equation must lie inside the unit circle.
4. AR processes produce spectra with sharp peaks (two poles of $A(z)$ per peak), whereas MA processes cannot produce peaky spectra.

ARMA modelling is a classic technique which has found a tremendous number of applications.

2.12 Summary: Wold's Decomposition Theorem and ARMA

- Every stationary time series can be represented as a sum of a perfectly predictable process and a feasible moving average process.
- Two time series with the same Wold representations are the same, as the Wold representation is unique.
- Since any MA process also has an ARMA representation, working with ARMA models is not an arbitrary choice but is physically justified.
- The causality and stationarity on ARMA processes depend entirely on the AR parameters and not on the MA parameters.
- An MA process is not uniquely determined by its ACF.
- An AR(p) process is always invertible, even if it is not stationary.
- An MA(q) process is always stationary, even if it is non-invertible.

Chapter 3

Intro to Estimation Theory

3.1 Introduction

- Historical data still allows for accurate modelling of complex systems.
- This links to a need for a unifying and rigorous framework that defines a 'goodness of performance' measure for any Data Analytic model

Terminology: **Prediction** refers to a pre-built model based on in-sample data to estimate out-of-sample values. **Forecasting** is a form of prediction that implicitly assumes time-series methods, and we use historical data to predict future data, often with probabilistic bounds (confidence intervals).

3.2 Statistical Estimation: Problem Statement

Estimators and Estimate **Definition 3.2.1**

Given an N-point dataset, $x[0], x[1], \ldots, x[N-1]$ that depends on unknown scalar parameter θ. We let $\hat{\theta}$ be an estimator that is defined as a function $g(\cdot)$ of dataset $\{x\}$, that can estimate our unknown parameter θ.

$$\hat{\theta} = g\big(x[0], x[1], \ldots, x[N-1]\big) \tag{3.1}$$

This is the single parameter or scalar case. We can consider the vector case, determining a set of parameters $\theta = [\theta_1, \ldots, \theta_p]^T$ from data samples $\mathbf{x} = [x[0], \ldots, x[N-1]]^T$ where such parameters will yield the highest probability of obtaining the observed data.

$$\max_{\text{span } \boldsymbol{\theta}} p(\mathbf{x}; \boldsymbol{\theta}) \tag{3.2}$$

The **estimator refers to our rule** $g(\mathbf{x})$ that assigns a value to parameter θ. And an **estimate** of the true value of θ is $\hat{\theta}$.

We can estimate θ with **classical estimation**, assuming θ is deterministic with no *a priori* information about it (minimum-variance solution, least-squares), or use **Bayesian estimation**, applying prior knowledge to it (Wiener and Kalman filters, adaptive signal processing).

$p(\mathbf{x}; \boldsymbol{\theta})$ contains all the information we need to find $\hat{\theta}$, however, in practice, the PDF is not given so we need to choose a model that captures the essence of the system we are trying to model – leading to a mathematically tractable estimator.

Gaussian RV **Definition 3.2.2**

A Gaussian random variable $X \sim N(\mu, \sigma^2)$ has the pdf

$$p_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.3}$$

μ is the mean, σ is the standard deviation, and is greater than zero. σ^2 is the variance.

Conditional PDF **Definition 3.2.3**

A conditional pdf $p_{y|x}(y \mid x)$ can be thought of as a sliced, normalised form of the joint pdf $p(x, y)$.

It is formally defined as

$$p_{Y|X}(y \mid x) = \begin{cases} \frac{p_{XY}(x,y)}{p_X(x)} & p_X(x) \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

General shorthand and notation means we define it as

$$p(x \mid y) = \begin{cases} \frac{p(x,y)}{p(y)} & p(y) \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

Bias **Definition 3.2.4**

The bias is the difference between the expected value of the estimate $\hat{\theta}$ and actual value of the parameter θ. It is denoted by B. For N data samples, we have

$$B = E\{\hat{\theta}_N - \theta\} \tag{3.6}$$

It can also be defined as $E[\eta]$, where $\eta = \hat{\theta} - \theta$.

3.3 Goodness of an Estimate

We assume noise $\mathbf{w}$ is white with i.i.d samples. But whiteness is not realistic, much rather, it is Gaussian. We can also assume it has zero-mean. These assumptions (later on) allow us to find a performance bound for optimal estimators, allowing us to gauge performance of an estimator.

Goodness analysis is usually a function of noise variance σ_w^2, usually expressed in terms of signal-to-noise ratio, SNR. Usually, we can only assess performance if we know true θ. Typically the goodness of an estimator is captured through the mean and variance of $\hat{\theta} = g(\mathbf{x})$.

We want $\mu_{\hat{\theta}} = E[\hat{\theta}] = \theta$, and a small variance, $\sigma_{\hat{\theta}} = E\{(\hat{\theta} - E\{\hat{\theta}\})^2\}$

Estimation error η is defined by $\eta = \hat{\theta} - \theta$. Since $\hat{\theta}$ is a random variable and θ is a constant, then η is a random variable too. Also, $\eta = 0$ refers to an unbiased estimator, which exactly matches the parameter θ.
We need:

1. $E\{\eta\} = 0$, an **unbiased** estimator
2. A minimum variance, where $\text{Var}(\eta) = E\{(\eta - E\{\eta\})^2\}$ is small

3.3.1 Bias

- For sufficiently large N observations of $x[n]$, the expected value of an unbiased estimate $\hat{\theta}$ converges towards its true value:

$$E\{\hat{\theta}_N\} = \theta \equiv B = E\{\hat{\theta}_N\} - \theta = 0 \tag{3.7}$$

- Conversely, if $B \neq 0$ then the estimator $\hat{\theta} = g(\mathbf{x})$ is biased asymptotically.
- Therefore it can be described that an estimator's bias is a critical measure of its **accuracy**. An unbiased estimator has an expected value equal to the parameter it estimates, while an asymptotically unbiased estimator approaches this property as the sample size grows infinitely large.

3.3.2 Variance

- The **precision** of an estimator is assessed by how closely it can consistently estimate the parameter. This is formalised by the variance of the estimator approaching zero as the sample size increases indefinitely, called

the **variance criterion**:

$$\lim_{N\to\infty} \text{var}(\hat{\theta}_N) = \lim_{N\to\infty} \left\{ E\left[\left| \hat{\theta}_N - E\{\hat{\theta}_N\} \right|^2 \right] \right\} = 0 \tag{3.8}$$

- This notion of precision can also be expressed in terms of the estimator's mean squared error (MSE) diminishing as the sample size grows:

$$\lim_{N\to\infty} \text{E}\left[(\hat{\theta}_N - \text{E}[\hat{\theta}_N])^2 \right] = 0 \tag{3.9}$$

  The Mean Square Convergence is a stronger form of converge than the aforementioned **variance criterion**.

- When the estimator is unbiased, which means its expected value is the true parameter value, we can apply Chebyshev's inequality to show that the estimator's probability of deviating from the true parameter by more than any positive number ϵ decreases as variance decreases:

$$\Pr\left\{ |\hat{\theta}_N - \theta| \geq \epsilon \right\} \leq \frac{\text{var}(\hat{\theta}_N)}{\epsilon^2} \tag{3.10}$$

- If the **variance of the estimator does indeed tend to zero as the sample size approaches infinity**, and the estimator is **asymptotically biased**, the estimator is **consistent**. This means that the probability that $\hat{\theta}$ differs from θ by more than ϵ will go to zero; it converges in probability to the true parameter value:

$$\hat{\theta}_N \xrightarrow{N\to\infty} \theta \tag{3.11}$$

Unbiased vs Consistent Estimators **Definition 3.3.1**

Unbiased estimators are defined by $\text{E}\{\hat{\theta}\} = \theta$. **Asymptotically unbiased estimators** are defined by $\text{E}\{\hat{\theta}_N\} = \theta$ when $n \to \infty$.

If an **asymptotically unbiased or unbiased estimator** has a variance that tends to zero as sample size grows to infinity, as shown by:

$$\lim_{N\to\infty} \left\{ \text{E}\left[\left| \hat{\theta}_N - \text{E}\{\hat{\theta}_N\} \right|^2 \right] \right\} = 0 \tag{3.12}$$

Then the estimator is **consistent**.

Tschebycheff (or Chebyshev) inequality ***Extra, Not Assessed* 3.3.1**

Fun trivia: Tschebycheff is one of many transliterations for Chebyshev, more include Tchebichef, Tchebychev, Tchebycheff, Tschebyschev...

Chebyshev's inequality provides a bound on the probability that a random variable deviates from its mean. To prove this, we will use Markov's inequality, which states that for a non-negative random variable X and any $a > 0$,

$$P(X \geq a) \leq \frac{\text{E}[X]}{a}. \tag{3.13}$$

Now let X be a random variable with finite mean μ and finite variance σ^2. For any $\epsilon > 0$, consider the non-negative random variable $(X - \mu)^2$. Applying Markov's inequality to this random variable, we have:

$$P((X-\mu)^2 \geq \epsilon^2) \leq \frac{\text{E}[(X-\mu)^2]}{\epsilon^2}. \tag{3.14}$$

Since the variance of X is σ^2, we have $\text{E}[(X-\mu)^2] = \sigma^2$. Therefore, we can rewrite the inequality as:

$$P((X-\mu)^2 \geq \epsilon^2) \leq \frac{\sigma^2}{\epsilon^2}. \tag{3.15}$$

Noticing that $(X-\mu)^2 \geq \epsilon^2$ if and only if $|X-\mu| \geq \epsilon$, we can write:

$$P(|X-\mu| \geq \epsilon) \leq \frac{\sigma^2}{\epsilon^2}. \tag{3.16}$$

This is the statement of Chebyshev's inequality. ■

3.3.3 Testing Estimators

Theoretical analysis helps measure the goodness (bias and variance) of the estimators, and use of simulations such as Monte Carlo help verify correctness of theoretical results, or give insight when we cannot find theoretical results.

Sometimes estimators have no optimality properties, but may perform good enough in practice.

3.4 Minimum Variance Unbiased (MVU) Estimation

In this section we aim to establish what are 'good' estimators of unknown deterministic parameters.
Denote A to be a deterministic signal that lies in interval $(-\infty, \infty)$, observed as $x[n]$ with additive white Gaussian noise (WGN) w:

$$x[n] = A + w[n] \qquad n = 0, 1, \ldots, N-1 \tag{3.17}$$

An example of an estimator for A would be the sample mean:

$$\hat{A} = \frac{1}{N} \sum_{n=0}^{N-1} x[n] \tag{3.18}$$

Another estimator for A would be half the sample mean:

$$\hat{\hat{A}} = \frac{1}{2N} \sum_{n=0}^{N-1} x[n] \tag{3.19}$$

But notice how $\mathrm{E}\{\hat{\hat{A}}\} = \begin{cases} 0 & A = 0 \\ A/2 & A \neq 0 \end{cases}$, making it parameter dependent! So it is not unbiased.

An biased estimator introduces a 'systemic error', which means for certain values of A, we will have to correct them, which makes things messy and inconvenient. Our goal is to avoid biased estimators.

3.4.1 Averaging Biased Estimators Example

See slides 31-32 for reference. Main takeaway: averaging biased estimators will not remove their bias.

3.4.2 Mean Square Error

The Mean Square Error (MSE), is defined as :

$$MSE(\hat{\theta}) = E\left\{(\hat{\theta} - \theta)^2\right\} \tag{3.20}$$

This measures the average mean squared deviation of the estimate $\hat{\theta}$ from the true value. We find that it is **equal to the variance of the estimator plus squared bias**.

$$\begin{aligned} MSE\left(\hat{\theta}\right) &= E\{(\hat{\theta} - \theta)^2\} = E\{\left[(\hat{\theta} - E\{\hat{\theta}\}) + (\underbrace{E\{\hat{\theta}\} - \theta}_{=bias,\ B(\hat{\theta})})\right]^2\} \\ &= E\left\{\left[\hat{\theta} - E\{\hat{\theta}\}\right]^2\right\} + 2B(\hat{\theta})\underbrace{E\{\hat{\theta} - E\{\hat{\theta}\}\}}_{=0} + B^2(\hat{\theta}) \\ &= var(\hat{\theta}) + B^2(\hat{\theta}) \end{aligned} \tag{3.21}$$

Chapter 4

Adaptive Estimation and Inference

4.1 Introduction

4.1.1 Objectives

- Introduce real-time adaptive estimation for **streaming** data
- Adaptive filters, ARMA models with adaptive coefficients, Wiener filter, Stochastic gradient algorithm, Least Mean Square (LMS)
- Learning rate, bias, variance
- Filtering configurations, nonlinear structures, stability and convergence, and applications

4.1.2 Adaptive Filters

- Adaptive filters: No assumptions of data
- Number guessing game: one person picks an integer within [-50,50]. The class tries to discover the number in the following styles of game:
  - Random guess with just correct/incorrect
  - Random guess with correct/incorrect and whether the number to be guessed was higher or lower
  - Nonstationarity: guessed number can fluctuate occasionally.
- This can be described as a recursive update:

$$g_i(n+1) = g_i(n) + \text{sign}(e(n)) \cdot \text{rand}[g_i(n), g_i(n-1)] \tag{4.1}$$

$$\text{new guess} = \text{old guess} + \text{correction} \tag{4.2}$$

The more formal way is a form of an adaptive filter:

$$\text{Correction Term} = \text{Learning Rate} \times \text{Function of Input Data} \times \text{Function of Output Error} \tag{4.3}$$

4.2 Wiener-Hopf Solution

4.2.1 Problem Formulation

Consider a vector of p sensor signals (thus representing a filter order p) with individual signals being $\mathbf{x(n)} = [x_1(n), \cdots, x_p(n)]^T$ weighted by the corresponding set of **time-varying** filter parameters $\mathbf{w}(n) = [w_1(n), \cdots, w_p(n)]^T$.

Let the output be the sum of weighted signals:

$$y(n) = \sum_{i=1}^{p} w_i(n)x_i(n) = \mathbf{x}^T(n)\mathbf{w}(n) \qquad n = 0, 1, 2, ... \tag{4.4}$$

Our objective is to find the **optimum** set of **fixed** weights $\mathbf{w}_o = [w_{o1}, \ldots, w_{op}]^T$ that minimises difference between system output and desired response $d(n)$. Let $d(n)$ be the desired response, or teaching signal, for the filter.

We can then define an error signal $e(n) = d(n) - y(n)$.

We will want to minimise the mean square error, or the expected value of the error power. Note that for convenience we use half of the expected value, it will have the same optimum weights but is neater to differentiate for other uses, such as finding the closed-form solution. Now, we define the error power, or the expected power, for a given weight, $J(\mathbf{w})$:

$$J(\mathbf{w}) = \frac{1}{2}E\{e^2(n)\} = \frac{1}{2}E\left\{\left(d(n) - \sum_{k=1}^{p} w_k(n)x_k(n)\right)^2\right\} \qquad \mathbf{w}(n) = [w_1(n), \cdots, w_p(n)]^T \tag{4.5}$$

$$= \frac{1}{2}E\{d^2\} - E\left\{\sum_{k=1}^{p} w_k x_k d\right\} + \frac{1}{2}E\left\{\sum_{j=1}^{p}\sum_{k=1}^{p} w_j w_k x_j x_k\right\} \tag{4.6}$$

$$= \frac{1}{2}E\{d^2\} - \sum_{k=1}^{p} w_k E\{x_k d\} + \frac{1}{2}\sum_{j=1}^{p}\sum_{k=1}^{p} w_j w_k E\{x_j x_k\} \tag{4.7}$$

Now, we introduce the following:

The variance of the desired signal, denoted σ_d^2, is defined as the expected value of d^2, representing the power of the teaching (desired) signal.

The cross-correlation between d and x_k is given by $r_{dx}(k)$, indicating the cross-correlation between d and x_k

The autocorrelation at lag $(j-k)$ for the stochastic process x is represented as $r_x(j,k)$, which expresses the autocorrelation at lag $(j-k)$:

$$\begin{aligned} \sigma_d^2 &= E\{d^2\} \\ r_{dx}(k) &= E\{dx_k\}, && k = 1, 2, \ldots, p \\ r_x(j,k) &= E\{x_j x_k\}, && j, k = 1, 2, \ldots, p \end{aligned}$$

Substitute the notation into Equation 4.7 and this yields:

$$J(\mathbf{w}) = \frac{1}{2}\sigma_d^2 - \sum_{k=1}^{p} w_k r_{dx}(k) + \frac{1}{2}\sum_{j=1}^{p}\sum_{k=1}^{p} w_j w_k r_x(j,k) \tag{4.8}$$

This is a multidimensional plot of cost function J against the weights, an error surface of the filter. This surface is bowl-shaped (see slide 7 page 10 for a 2-parameter example) and has a global minimum point that is well-defined.
The **Wiener solution** follows a least-squares approach. Denote the optimum value of weight w_k as w_{ok}.

$$\nabla_{w_k} J = \frac{\partial J}{\partial w_k} = \frac{\partial}{\partial w_k}\left[\frac{1}{2}\sigma_d^2 - \sum_{k=1}^{p} w_k r_{dx}(k) + \frac{1}{2}\sum_{j=1}^{p}\sum_{k=1}^{p} w_j w_k r_x(j,k)\right] \quad k = 1, \ldots, p \tag{4.9}$$

$$= -r_{dx}(k) + \sum_{j=1}^{p} w_j r_x(j,k) = 0 \tag{4.10}$$

$$= \sum_{j=1}^{p} w_{oj} r_x(j,k) = r_{dx}(k), \quad k = 1, 2, \ldots, p \quad \Leftrightarrow \quad \mathbf{R}_{xx}\mathbf{w}_o = \mathbf{r}_{d\mathbf{x}} \tag{4.11}$$

$$\Rightarrow \mathbf{w}_o = \mathbf{R}_{xx}^{-1}\mathbf{r}_{dx} \tag{4.12}$$

The system of equations we are left with are termed the Wiener-Hopf equations, where a filter with weights that satisfy the equations is called a Wiener filter. Notice this is a block filter that operates on the whole set of data, making it non-sequential. This will not work for streaming data.
It is also computationally demanding to calculate inverses for large correlation matrices $\mathbf{R}_{xx}$.

4.2.2 Method of steepest descent

This is an iterative Wiener solution that allows weights to be adjusted in an iterative fashion along the error surface. Take the direction opposite to the gradient vector, with elements defined by $\nabla_{w_k} J$ for $k = 1, 2, \cdots, p$. Assume a teaching signal is $d(n) = \mathbf{x}^T(n)\mathbf{w}_0 + q(n)$, where q is a noise term following a normal distribution $q \sim \mathcal{N}(0, \sigma_q^2)$, so that we have $J_{\min} = \sigma_q^2$.

We have

$$\nabla_{w_k} J(n) = -r_{dx}(k) + \sum_{j=1}^{p} w_j(n) r_x(j,k) \tag{4.13}$$

The adjustment applied to weight $w_k(n)$ at iteration n, called the weight update $\Delta w_k(n)$, is defined as the negative, opposing direction of the gradient:

$$\Delta w_k(n) = -\mu \nabla_{w_k} J(n), \quad k = 1, 2, \ldots, p \tag{4.14}$$

where μ is a small positive constant, $\mu \in \mathbb{R}^+$, called the learning rate parameter (also called step size, usually denoted by μ or η).

Recall the gradient of J with respect to the k-th weight:

$$\nabla_{w_k} J(n) = -r_{dx}(k) + \sum_{j=1}^{p} w_j(n) r_x(j,k) \quad \text{or} \quad \nabla_{\mathbf{w}} J(n) = -\mathbf{r}_{dx} + \mathbf{R}\mathbf{w} \tag{4.15}$$

Given the *current* value of the k-th weight $w_k(n)$ at iteration n, the *updated* value of this weight at the next iteration $n+1$ is computed as:

$$w_k(n+1) = w_k(n) + \Delta w_k(n) = w_k(n) - \mu \nabla_{w_k} J(n) \tag{4.16}$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \Delta \mathbf{w}(n) = \mathbf{w}(n) - \mu \nabla_{\mathbf{w}} J(n) \tag{4.17}$$

where μ is a small positive constant, representing the learning rate. The **updated filter weights** are equal to the **current weights** plus the **weight update**. We then have:

$$w_k(n+1) = w_k(n) + \mu \left[r_{dx}(k) - \sum_{j=1}^{p} w_j(n) r_x(j,k) \right], \quad k = 1, \ldots, p \tag{4.18}$$

or in a vector form:

Steepest Descent Update Rule — **Definition 4.2.1**

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \left[\mathbf{r}_{dx} - \mathbf{R}\mathbf{w}(n) \right] \tag{4.19}$$

The SD method is **exact** in that no approximations are made in the derivation; the key difference is that the solution is obtained iteratively. Observe that there is no matrix inverse in the update of filter weights!
Notice we have an adaptive parameter estimator that is determined in an iterative form, i.e. new parameter estimate = old parameter estimate + update.

- A **spatial filter**'s cost function is an ensemble average taken at instant n over an ensemble of spatial filters. An example could be nodes in a sensor array.
- A temporal filter's cost function is the is the sum of error squares over time, where ACF and other processes use time averages. If the processes are jointly ergodic it is justified to substitute time averages for ensemble averages. The sum of error squares is computed by:

$$\mathcal{E}_{total} = \sum_{i=1}^{n} \mathcal{E}(i) = \frac{1}{2} \sum_{i=1}^{n} e^2(i) \tag{4.20}$$

4.2.3 Learning Rate or Step Size

Choosing the right learning rate μ is crucial to converge on the global minimum solution.

- For μ that is small enough the steepest descent (SD) algorithm will converge to a stationary point (local or global minimum) where $J(e) \equiv J(\mathbf{w}_o)$ where $\nabla_w J((w)_o = 0$
- When μ is small compared to critical value $\mu)crit$, the trajectory by the weight vector $\mathbf{w}(n)$ for increasing n tends to be monotonic (steady movement towards solution).
- But as μ gets closer to μ_{crit}, the trajectory is oscillatory or overdamped.
- When μ exceeds μ_{crit}, trajectory becomes unstable.

4.2.4 Matrix Formulation

Similar to the previous section, the vector-matrix formulation of the Wiener filter is as follows:

The cost (error objective) function, J, defined as $J = \frac{1}{2}E\{\mathbf{e}^2(n)\}$, can be expanded by:

$$\begin{aligned} J &= \frac{1}{2}E\{\mathbf{e}\mathbf{e}^T\} \\ &= \frac{1}{2}E\{(\mathbf{d} - \mathbf{w}^T\mathbf{x})(\mathbf{d} - \mathbf{w}^T\mathbf{x})^T\} \\ &= \frac{1}{2}E\{\mathbf{d}^2 - \mathbf{d}\mathbf{x}^T\mathbf{w} - \mathbf{w}^T\mathbf{x}\mathbf{d}^T + \mathbf{w}^T\mathbf{x}\mathbf{x}^T\mathbf{w}\} \\ &= \frac{1}{2}E\{\mathbf{d}^2 - 2\mathbf{d}\mathbf{x}^T\mathbf{w} + \mathbf{w}^T\mathbf{x}\mathbf{x}^T\mathbf{w}\} \\ &= \frac{1}{2}E\{\mathbf{d}^2\} - \mathbf{w}^T E\{\mathbf{d}\mathbf{x}^T\} + \frac{1}{2}\mathbf{w}^T E\{\mathbf{x}\mathbf{x}^T\}\mathbf{w} \end{aligned} \tag{4.21}$$

where the cross-correlation vector $\mathbf{r}_{dx} = E\{\mathbf{d}\mathbf{x}^T\}$ and the autocorrelation matrix $\mathbf{R} = E\{\mathbf{x}\mathbf{x}^T\}$.
Thus, with $\mathbf{w}$ being still a fixed vector for the time being, the cost function J can be expressed as:

$$J = \frac{1}{2}\sigma_d^2 - \mathbf{w}^T\mathbf{r}_{dx} + \frac{1}{2}\mathbf{w}^T\mathbf{R}\mathbf{w} \tag{4.22}$$

This function is quadratic in $\mathbf{w}$ and, for a full-rank $\mathbf{R}$, it has a unique minimum, $J(\mathbf{w}_0)$.

For $J_{min} = J(\mathbf{w}_0)$ implies that the derivative of J with respect to $\mathbf{w}$, $\frac{\partial J}{\partial \mathbf{w}} = -\mathbf{r}_{dx} + \mathbf{R}\cdot\mathbf{w} = \mathbf{0}$, which results in:

$$\mathbf{0} = -\mathbf{r}_{dx} + \mathbf{R}\cdot\mathbf{w}_0 \tag{4.23}$$

Finally, we obtain the Wiener-Hopf equation:

$$\mathbf{w}_0 = \mathbf{R}^{-1}\mathbf{r}_{dx} \tag{4.24}$$

4.3 Least Mean Square (LMS) Algorithm

Recall: from the steepest descent (SD) rule, we have:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\left[\mathbf{r}_{dx} - \mathbf{R}\mathbf{w}(n)\right] \tag{4.25}$$

However correlations $\mathbf{r}_{dx}, \mathbf{R}$ are not readily available or time-consuming to complete. The Least Mean Squares minimises $\mathbf{J}(\mathbf{n}) = \frac{1}{2}e^2(n)$ using instant estimates of the following processes:

$$\hat{\mathbf{R}}(n) = \mathbf{x}(n)\mathbf{x}^T(n) \tag{4.26}$$

$$\hat{\mathbf{r}}_{d\mathbf{x}}(n) = d(n)\mathbf{x}(n) \tag{4.27}$$

We then substitute into the SD equation:

$$\begin{aligned} \mathbf{w}(n+1) &= \mathbf{w}(n) + \mu\big[d(n)\mathbf{x}(n) - \mathbf{x}(n)\underbrace{\mathbf{x}^T(n)\mathbf{w}(n)}_{y(n)}\big] \\ &= \mathbf{w}(n) + \mu[\underbrace{d(n) - y(n)}_{e(n)}]\mathbf{x}(n) \end{aligned} \tag{4.28}$$

We are then left with:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\mathbf{x}(n) \tag{4.29}$$

Generally, LMS takes longer to converge and follows a jagged 'zig-zag' pattern, however it is computationally much cheaper to implement.

4.3.1 Temporal Problems Example with LMS

4.3.2 Visualisations of convergence

- Chapter, Slide 23
- Chapter 7, Slide 27

4.3.3 Convergence of LMS vs MVU Estimation

- A convergence in the mean implies a bias in parameter estimation. $E\{\mathbf{w}(n)\} \to \mathbf{w}_0 \quad \text{as } n \to \infty$, similar to the requirement for an unbiased optimal weight estimate.
- Convergence in the mean square error implies a variance in the estimator, as it fluctuates around the instant weight vector estimates. Since error is a function of filter weights, it can be denoted $E\{e^2(n)\} \to$ constant as $n \to \infty$
- We expect MSE convergence condition to be tighter. If LMS is not convergent in mean square error, then it is convergent in mean, but converse is not always true. (i.e. if an estimator is unbiased, it is not necessarily minimum variance, but if it is minimum variance it is likely unbiased.)

4.4 Applications of adaptive filters

- **Forward prediction:** desired signal is input signal advanced relative to input of adaptive filter. Examples: financial forecasting, wind prediction
- **System identification:** adaptive filter and unknown system are connected in parallel and fed same input signal $x(n)$. Examples: echo cancellation, removing feedback whistling in teleconferencing, hearing aids
- **Inverse system modelling:** adaptive filter cascaded with unknown system. Examples: channel equalisation in mobile telephony, wireless sensor networks, underwater communications, mobile sonar, mobile radar
- **Noise Cancellation:** assuming noise in primary input and reference noise are correlated. Noise removal in phones, concert halls, video recording

4.5 Elements of Neural Networks

4.5.1 Motivations of nonlinear structures

- Cannot separate signals with overlapping components without nonlinear models
- Cannot capture nonlinear signals that are non-gaussian

4.5.2 Model of an artificial neuron

- Delayed inputs x
- Bias input with unity value
- Takes the sums and multilies
- Nonlinear output (activation function)

Nonlinear output: distorts/attenuates the signal more at the extremities. Bounded input, bounded output. NNs represent nonlinear maps from one metric space to another