Introduction to Optimal Wiener Filters

1 Introduction & Setup

There are many setting where we have an input sequence $\{x[n]\}$ that consists of a desired input signal $\{s[n]\}$ which has been corrupted with undesirable additive noise $\{w[n]\}$. The goal, which is a fundamental principal in signal processing, is to filter the input signal such that the noise and interference is ameliorated from the output of the system responsible for filtering the signal. In todays lecture we shall treat the problem of filtering such a signal by estimating the presence of the additive noise. This estimation is constrained to a linear filter with an impulse response given by $\{h[n]\}$. The output of the filter produces the desired signal $\{d[n]\}$. Figure 1 shows the problem of this estimation using a linear filter.

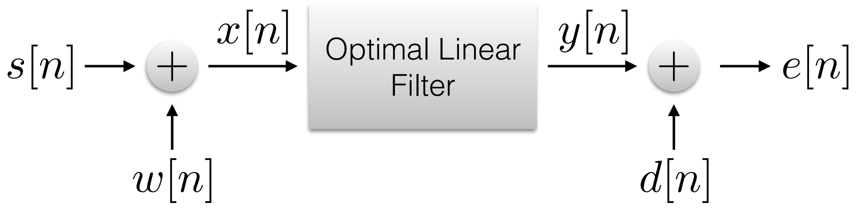

Figure 1: Model for a linear estimation problem.

The input signal $x[n] = s[n] + w[n]$ is presented to a linear filter with the output being given by $y[n]$. The error from this estimation is given by $e[n] = d[n] - y[n]$. There are three specific case to consider.

- If $d[n] = s[n]$ then the linear estimation problem is referred to as *filtering*.
- If $d[n] = s[n + D]$, where $D > 0$ the problem is known as signal *prediction*.
- If $d[n] = s[n - D]$, where $D > 0$ the problem is known as signal *smoothing*.

In this lecture we focus on signal filtering and prediction. One of the central issues that we should make clear is that the user designing the filter must know the precise spectrum of the input domain to develop a precise filter. However, it is rarely the situation that the user is going to know all of these characteristics prior to building a filter. Often, we specify the filter characteristics based on some past experience, and/or trial and error. FIR/IIR design techniques can design any filter you specify – it does not tell you whether that is the right filter for that specific signal. Our discussion of Wiener filters with present an approach to handling the situation where we do not have such prior knowledge. The *Wiener filter* is one such that it achieves the minimum mean-squared error (MMSE).

2 FIR Wiener Filters

2.1 Formulation and Derivation

Let us begin the discussion of Wiener filters from an FIR filter, whose filter is of length M with coefficients $h[k]$ for $k \in \{1, \dots, M\}$. The output of the filter $y[n]$ is the convolution of x with h, which is given by:

$$y[n] = \sum_{k=0}^{M-1} h[k]x[n-k] \tag{1}$$

Equation (1) looks simply as an FIR filter, however, what makes the Weiner filter unique is how we determine the coefficients. Hence the output $y[n]$ depends on a sequence on the finite data $x[n], x[n-1], \dots, x[n-M+1]$. The error of the difference between the desired output $d[n]$ and y[n] is given by:

$$\begin{aligned}\epsilon_M &= \mathbb{E}\left[|e[n]|^2\right] \\ &= \mathbb{E}\left[\left|d[n] - \sum_{k=0}^{M-1} h[k]x[n-k]\right|^2\right] \qquad (2)\\ &= \mathbb{E}\left[\left|d[n] - \mathbf{h}^H\mathbf{x}_{n:n-M+1}\right|^2\right] \qquad (3)\end{aligned}$$

A Note on Orthogonality: Let us estimate a signal $s[n]$ using $y[n] = \mathbf{h}^H\mathbf{x}_{n:n-M+1}$ where the filter coefficients $\mathbf{h}$ minimize the mean-squared error.

$$\sigma^2_{\text{err}} = \mathbb{E}\left[s[n] - \mathbf{h}^H\mathbf{x}_{n:n-M+1}\right]$$

If $\epsilon = s[n] - \mathbf{h}^H\mathbf{x}_{n:n-M+1}$ then $\mathbf{h}$ minimizes σ^2_{err} is $\mathbf{h}$ is chosen such that:

$$\mathbb{E}\left[e[n]x^*[n-l]\right] = \mathbb{E}\left[x^*[n-l]e[n]\right]$$

for $l = 1, \dots, M$. That is to say the error is orthogonal to the input sequence used to compute the filter output.

$$\begin{aligned}\mathbb{E}\left[x[n-l]e^*[n]\right] &= \mathbb{E}\left[x[n-l]\left(d[n] - \sum_{k=0}^{M-1} h[k]x[n-k]\right)\right] \\ &= \mathbb{E}\left[x[n-l]d[n] - \sum_{k=0}^{M-1} h[k]x[n-l]x[n-k]\right] \\ &= \mathbb{E}\left[x[n-l]d[n]\right] - \mathbb{E}\left[\sum_{k=0}^{M-1} h[k]x[n-l]x[n-k]\right] \\ &= \mathbb{E}\left[x[n-l]d[n]\right] - \sum_{k=0}^{M-1} h[k]\mathbb{E}\left[x[n-l]x[n-k]\right] \\ &= 0\end{aligned}$$

The error in Equation (2) is a quadratic function w.r.t. the filter coefficients, hence the minimization of ϵ_M results in a set of linear equations given by:

$$\sum_{k=0}^{M-1} h[k]\gamma_{xx}[l-k] = \gamma_{dx}[l] \tag{4}$$

where $\gamma_{xx}[k]$ is the autocorrelation of the input sequence $\{x[n]\}$ and $\gamma_{dx}[l] = \mathbb{E}[d[n]x^*[n-k]]$ is the crosscorrelation between the desired sequence and the input sequence. This set of linear equations is known as the *Wiener-Hopf equation*. In general, Equation (4) can be written in matrix form which is expressed as:

$$\mathbf{\Gamma}_M \mathbf{h}_M = \boldsymbol{\gamma}_d$$

where $\mathbf{\Gamma}_M$ is an $M \times M$ (Hermitian) Toeplitz matrix with the elements $\mathbf{\Gamma}_M(l,k) = \gamma_{xx}[l-k]$ and $\boldsymbol{\gamma}_d$ is a crosscorrelation vector. The solution is, hence, given by:

$$\mathbf{h}_{\text{opt}} = \mathbf{\Gamma}_M^{-1}\boldsymbol{\gamma}_d \tag{5}$$

which results in a minimum error of:

$$\begin{aligned}\text{MMSE}_M = \min_{\mathbf{h}_M} \epsilon_M &= \sigma_d^2 - \sum_{k=0}^{M-1} h_{\text{opt}}[k]\gamma_{dx}^*[k] \\ &= \sigma_d^2 - \boldsymbol{\gamma}_d^{*\mathsf{T}}\mathbf{\Gamma}_M^{-1}\boldsymbol{\gamma}_d\end{aligned}$$

where $\sigma_d^2 = \mathbb{E}[|d[n]|^2]$.

As stated earlier the correlation matrix being inverted is Toeplitz and the generalized Levinson-Durbin algorithm can be used to solve for the optimal filter coefficients.

2.2 Notes on Implementing a Wiener Filter

Solving Equation (5) appears to be very straight forward, especially since we have learned about the Levinson-Durbin algorithm in the last class. However, there are some very serious concerns with this approach. First, autocorrelation matrix can be near singular (it cannot be completely singular, why not?), in which case the inverse will be unreliable. can check the condition of the matrix in Matlab using `c=cond(`$\mathbf{\Gamma}_M$`)`. If `c` $> 10^4$ then the matrix is *near* singular. If the matrix is near singular than you can reduce M. Second, even if the matrix is not near-singular, for large matrices, the inversion may take a very long time, though the Levinson-Durbin algorithm helps reduce the complex of computing the inverse with $\mathcal{O}(M^3)$ operations (using Gauss-Jordan eliminations).

References

- J. G. Proakis and D. G. Manolakis, "Digital Signal Processing: Principles, Algorithms, and Applications," Prentice Hall, 4th Ed., 2007.