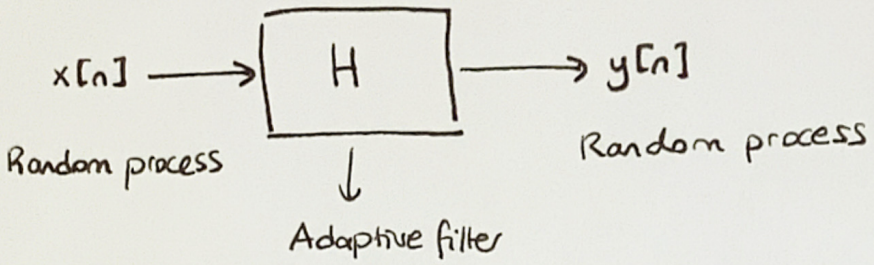
12/03/20

Today's Lecture

* Wiener filters
* Linear minimum mean-square error (MSE) filter design
  - MMSE filter using Wiener-Hopf equations
* One-step forward predictor
* Levinson-Durbin recursion to determine filter coefficients for a deterministic autocorrelation matrix

Wiener Filters



Design H to drive y[n] to a desired output d[n].

Wiener filter optimizes the mean square error (MSE) between y[n] and d[n]:

$$E\left[(d[n] - y[n])^2 \right]$$

x[n] and y[n] are random processes

H is an LTI filter,

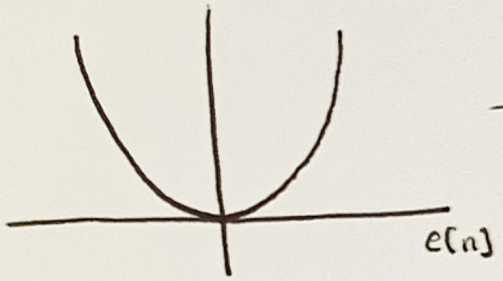
Performance criterion: minimum mean square error (MMSE)

$$y[n] = \sum_{i=0}^{\infty} h[i]\, x[n-i]$$ IIR causal filter

Assumptions: $x[n], d[n]$ are 0 mean, WSS (autocorrelation is stationary)

We want to minimize the mean square error

$$E[e^2[n]] = E[(d[n]-y[n])^2]$$



→ convex function of $h[k]$: $\frac{\partial^2 E[e^2[n]]}{\partial h[k]^2} \geq 0$ (will show this.)

* to minimize $E[e^2[n]]$ as a function of $h[k]$, we take its derivative (gradient) and set it to 0.

$$\frac{\partial E[e^2[n]]}{\partial h[k]} = E\left[2e[n]\frac{\partial e[n]}{\partial h[k]}\right]$$

where $e[n] = d[n] - \sum_{i=0}^{\infty} h[i]\, x[n-i]$

$$\frac{\partial e[n]}{\partial h[k]} = -x[n-k]$$

$$\Rightarrow \frac{\partial E[e^2[n]]}{\partial h[k]} = E[-2e[n]x[n-k]] = -2E[e[n]x[n-k]]$$

$$= 0 \quad \text{for optimality.}$$

Condition for optimality: $e[n]$ is orthogonal to all inputs $x[n], x[n-1], \ldots$

$$\frac{\partial E[e^2[n]]}{\partial h[k]} = -2E\left[\left(d[n] - \sum_{i=0}^{\infty} h[i]x[n-i]\right)x[n-k]\right] = 0$$

$$E[d[n]x[n-k]] - E\left[\sum_{i=0}^{\infty} h[i]x[n-i]x[n-k]\right] = 0$$

$$E[d[n]x[n-k]] - \sum_{i=0}^{\infty} \underbrace{E[x[n-i]x[n-k]]}_{r[i-k]} h[i] = 0$$

$$\Rightarrow \sum_{i=0}^{\infty} h[i]r[i-k] = E[d[n]x[n-k]] \triangleq p[-k]$$

Wiener-Hopf Equations

$$\sum_{i=0}^{\infty} h[i]\,r[i-k] = p[-k]$$

$r[i-k]$: autocorrelation; $p[-k]$: cross correlation

If we only consider a length-N FIR filter, then the Wiener-Hopf equations become

$$\sum_{i=0}^{N-1} h[i]\,r[i-k] = p[-k] \quad , \quad k = 0, \ldots, N-1$$

$$\begin{bmatrix} r[0] & r[1] & \cdots & r[N-1] \\ r[1] & r[0] & \cdots & r[N-2] \\ \vdots & & \ddots & \vdots \\ r[N-1] & r[N-2] & \cdots & r[0] \end{bmatrix} \begin{bmatrix} h[0] \\ h[1] \\ \vdots \\ h[N-1] \end{bmatrix} = \begin{bmatrix} p[0] \\ p[-1] \\ \vdots \\ p[-(N-1)] \end{bmatrix}$$

NXN autocorrelation matrix; NX1 vector; NX1 vector

$$R \cdot \hat{h} = p$$

p → cross correlation vector

$\hat{h}$ → optimal filter coefficients (taps)

$$\boxed{\hat{h} = R^{-1} p}$$ (R\p in MATLAB) or 'inv'

* What is the minimum error we can achieve?

$$E[e^2[n]] = E\left[\underbrace{\left(d[n] - \sum_{k=0}^{N-1} h[k]x[n-k]\right)}_{e[n]}\left(d[n] - \sum_{l=0}^{N-1} h[l]x[n-l]\right)\right]$$

$$= E[d^2[n]] - 2\sum_{k=0}^{N-1} h[k]\,E[d[n]x[n-k]]$$

$$+ \sum_{k=0}^{N-1}\sum_{l=0}^{N-1} h[k]h[l]\,E[x[n-k]x[n-l]]$$

Note that $d[n]$ is zero-mean $\rightarrow$ $E[d^2[n]] = Var(d[n]) = \sigma_d^2$

$E[d[n]x[n-k]] = p[-k]$ (cross-correlation)

$E[x[n-k]x[n-l]] = r[k-l]$ (auto-correlation)

$$\Rightarrow E[e^2[n]] = \sigma_d^2 - 2\sum_{k=0}^{N-1} h[k]p[-k] + \sum_{k=0}^{N-1}\sum_{l=0}^{N-1} h[k]h[l]r[k-l]$$

$$= \sigma_d^2 - 2h^T p + h^T R h \quad \text{verify}$$

Claim: At optimal h,

$$\hat{h} = R^{-1}p$$

$(AB)^T = B^T A^T$

$$E[e^2[n]] = \sigma_d^2 - 2(R^{-1}p)^T p + (R^{-1}p)^T R (R^{-1}p)$$

$$= \sigma_d^2 - 2p^T R^{-1} p + \underbrace{p^T \underbrace{R^{-1} R}_{I} R^{-1} p}_{p^T R^{-1} p}$$

(Note: $R^T = R$, $(R^{-1})^T = R^{-1}$)

$$= \sigma_d^2 - p^T R^{-1} p$$

We can show that the error for general h satisfies

$$E[e^2[n]] = \underbrace{(\sigma_d^2 - p^T R^{-1} p)}_{\text{error for optimal } h = \hat{h}} + \underbrace{p^T R^{-1} p - 2h^T p + h^T R h}_{\geq 0}$$

Note that $p = R\hat{h}$. Then

$$\begin{aligned} p^T R^{-1} p - 2h^T p + h^T R h &= \hat{h}^T R R^{-1} R \hat{h} - 2h^T R \hat{h} + h^T R h \\ &= \hat{h}^T R \hat{h} - 2h^T R \hat{h} + h^T R h \\ &= (h - \hat{h})^T R (h - \hat{h}) \geq 0 \end{aligned}$$

Since R is positive semi definite

$x^T R x \geq 0$ for any x.

Why $E[e^2[n]]$ is convex function of $h[k]$?

$$\frac{\partial E[e^2[n]]}{\partial h[k]} = E[-2e[n]x[n-k]]$$

$$\frac{\partial^2 E[e^2[n]]}{\partial h[k]^2} = E\left[-2\frac{\partial e[n]}{\partial h[k]} x[n-k] - 2e[n]\underbrace{\frac{\partial x[n-k]}{\partial h[k]}}_{0}\right]$$

$$\frac{\partial e[n]}{\partial h[k]} = -x[n-k]$$

$$= E[-2(-x[n-k])x[n-k]]$$

$$= 2E[x^2[n-k]] \geq 0$$

Optimal filter $\hat{h}$ is unique because mean square error is a quadratic function (convexity).

$$\hat{h} = R^{-1} p.$$

Important special case:

Linear prediction: We want to predict $x[n]$ from $x[n-1], x[n-2], \cdots, x[n-N]$.

One-step-forward linear predictor

$$\hat{x}[n] = \sum_{k=1}^{N} h[k]\, x[n-k] \quad \text{where} \quad h[k]\text{'s are optimized to minimize MSE.}$$

$$\hat{x}[n+1] = \sum_{k=1}^{N} h[k]\, x[n+1-k] = \sum_{k=0}^{N-1} h[k+1]\, h[n-k]$$

This corresponds to $d[n] = x[n+1]$

If

$d[n] = x[n]$ filtering problem

$d[n] = x[n+k]$, $k>0$ prediction problem

$d[n] = x[n-k]$, $k>0$ smoothing problem (e.g., AR process)

For one-step-forward prediction, we can find the solution with a Wiener filter:

$$d[n] = x[n+1]$$

$$e[n] = x[n+1] - \hat{x}[n+1]$$

Minimize $E[e^2[n]]$

$$R\hat{h} = p$$

$N \times N$ $N \times 1$ $N \times 1$

$$p[-k] = E\left[x[n-k]\, \overbrace{x[n+1]}^{d[n]}\right], \quad k = 0, \cdots, N-1$$

$x[n-k]$: input signal; $d[n]$: desired signal

$$p = \begin{bmatrix} E[x[n]x[n+1]] \\ E[x[n-1]x[n+1]] \\ \vdots \\ E[x[n-(N-1)]x[n+1]] \end{bmatrix}$$

$$\begin{bmatrix} r[0] & r[1] & \cdots & r[N-1] \\ r[1] & r[0] & \cdots & r[N-2] \\ \vdots & & & \vdots \\ r[N-1] & r[N-2] & \cdots & r[0] \end{bmatrix} \begin{bmatrix} h[0] \\ h[1] \\ \vdots \\ h[N-1] \end{bmatrix} = \begin{bmatrix} r[1] \\ r[2] \\ \vdots \\ r[N] \end{bmatrix}$$

$r[1] \rightarrow p[0] = E[x[n]x[n+1]]$

$r[2] \rightarrow p[-1]$

$r[N] \rightarrow p[-(N-1)] = E[x[n-(N-1)]\, x[n+1]]$

Wiener-Hopf for one-step-forward prediction

Error for the optimal $\hat{h}$:

MMSE : $E[e^2[n]] = \sigma_d^2 - p^T R^{-1} p = e$ (constant)

where $\sigma_d^2 = E[d^2[n]] = E[x^2[n+1]] = r(0)$ WSS

$\Rightarrow e = r(0) - p^T \hat{h}$ since $R\hat{h} = p$, $\hat{h} = R^{-1}p$

$$\begin{matrix} \text{row } 1 \rightarrow \\ \text{rows } 2\text{-}N+1 \rightarrow \end{matrix} \begin{bmatrix} r(0) & p^T \\ p & R \end{bmatrix} \underbrace{\begin{bmatrix} 1 \\ -\hat{h} \end{bmatrix}}_{a_N} = \begin{bmatrix} e \\ 0 \end{bmatrix} \begin{matrix} \rightarrow \text{constant} \\ \rightarrow N \times 1 \text{ vector} \end{matrix}$$

$(N+1) \times (N+1)$

$$a_N = \begin{bmatrix} (a_N)_0 \\ (a_N)_1 \\ \vdots \\ (a_N)_N \end{bmatrix}$$

$$\begin{bmatrix} r(0) & r(1) & \cdots & r(N) \\ r(1) & r(0) & \cdots & r(N-1) \\ \vdots & & \ddots & \vdots \\ r(N) & r(N-1) & \cdots & r(0) \end{bmatrix} \begin{bmatrix} (a_N)_0 \\ (a_N)_1 \\ \vdots \\ (a_N)_N \end{bmatrix} = \begin{bmatrix} e \\ 0 \\ \vdots \\ 0 \end{bmatrix} \Rightarrow \sum_{l=0}^{N} (a_N)_l r(l) = e$$

$$\sum_{l=0}^{N} (a_N)_l r(l-i) = 0 \quad \text{for } i = 1, \cdots, N$$

$(N+1) \times (N+1)$

The Levinson-Durbin Algorithm

A procedure to recursively calculate the solution to an equation involving a Toeplitz matrix (R is Toeplitz)

Algorithm runs in $O(n^2)$ time, which is a strong improvement over Gauss-Jordan elimination $O(n^3)$.

Question: How can we take the optimal length-N predictor and easily determine the optimal length-$N+1$ predictor?

$$a_N = \begin{bmatrix} 1 \\ -\hat{h} \end{bmatrix}$$

- If we choose $d[n] = x[n-M]$ and predict this from the samples $x[n]$, $x[n-1], \ldots, x[n-(M-1)]$, this would be a similar problem (smoothing or backwards prediction)

- Wiener-Hopf in this case gives a similar-looking system for the optimal filter $g[1], \ldots, g[N]$.

Backwards prediction

$$\begin{bmatrix} r[0] & r[1] & \cdots & r[N-1] \\ r[1] & r[0] & \cdots & r[N-2] \\ \vdots & & & \vdots \\ r[N-1] & r[N-2] & \cdots & r[0] \end{bmatrix} \begin{bmatrix} g(1) \\ g(2) \\ \vdots \\ g(N) \end{bmatrix} = \begin{bmatrix} p(N) \\ p(N-1) \\ \vdots \\ p(1) \end{bmatrix}$$

$\Rightarrow$ The optimal g's are just the flipped versions of optimal h's.

$$\hat{g}[k] = \hat{h}[(N+1)-k]$$

The best expected MSE (MMSE) is exactly the same.

The optimal a_N can be updated from the optimal a_{N-1} as follows:

$$a_N = \begin{bmatrix} a_{N-1} \\ 0 \end{bmatrix} + \Gamma_N \begin{bmatrix} 0 \\ a_{N-1}^B \end{bmatrix} \qquad (N+1) \times 1 \text{ vector}$$

- $\begin{bmatrix} a_{N-1} \\ 0 \end{bmatrix}$: best forward prediction
- Γ_N: scalar reflection coefficient
- $\begin{bmatrix} 0 \\ a_{N-1}^B \end{bmatrix}$: flipped one version of a_{N-1} (best backwards prediction)

In scalar form,

$$(a_N)_k = (a_{N-1})_k + \Gamma_N (a_{N-1})_{N-k} \quad , \; k = 0, \cdots, N$$

$$(a_{N-1})_0 = 1$$

$$(a_{N-1})_N = 0$$

* How can we calculate Γ_N?

$$R_{N+1}\, a_N = R_{N+1} \begin{bmatrix} a_{N-1} \\ 0 \end{bmatrix} + R_{N+1}\, \Gamma_N \begin{bmatrix} 0 \\ a_{N-1}^B \end{bmatrix}$$

- R_{N+1}: $(N+1) \times (N+1)$ autocorrelation matrix
- I: $R_{N+1} \begin{bmatrix} a_{N-1} \\ 0 \end{bmatrix}$
- II: $R_{N+1}\, \Gamma_N \begin{bmatrix} 0 \\ a_{N-1}^B \end{bmatrix}$

$$\begin{bmatrix} r(0) & r^T \\ r & R_N \end{bmatrix} \begin{bmatrix} 1 \\ -\hat{h}_N \end{bmatrix} = \begin{bmatrix} \hat{e}_N \\ \underline{0} \end{bmatrix}$$

$\hat{e}_N$: scalar best error (MMSE) with N taps

$\underline{0}$: length N $\underline{0}$ vector

I. $$R_{N+1} \begin{bmatrix} a_{N-1} \\ 0 \end{bmatrix} = \begin{bmatrix} R_N & r_N^B \\ (r_N^B)^T & r(0) \end{bmatrix} \begin{bmatrix} a_{N-1} \\ 0 \end{bmatrix} = \begin{bmatrix} R_N\, a_{N-1} \\ (r_N^B)^T a_{N-1} \end{bmatrix} = \begin{bmatrix} \hat{e}_{N-1} \\ \underline{0} \\ \Delta_{N-1} \end{bmatrix}$$

$\hat{e}_{N-1}$: scalar; $\underline{0}$: vector; Δ_{N-1}: scalar

where

$$\sum_{l=0}^{N-1} (a_{N-1})_l\, r(l) = \hat{e}_{N-1}$$

$$\sum_{l=0}^{N-1} (a_{N-1})_l\, r(l-i) = 0 \quad , \; i = 1, \cdots, N$$