12/07/20

Today's Lecture

* Levinson - Durbin recursion
* Steepest descent algorithm
* Least Mean Square (LMS)
* Least Squares (LS)

## The Levinson - Durbin Algorithm

$$R_{N+1}\, a_N = \begin{bmatrix} \hat{e}_N \\ \underline{0} \end{bmatrix} \qquad \text{where} \quad a_N = \begin{bmatrix} 1 \\ -\hat{h} \end{bmatrix}, \quad \hat{h} = R_N^{-1} p$$

$$p = [r(1)\ r(2) \dots r(N)]^T$$

The optimal $a_N$ is obtained from the optimal $a_{N-1}$ as

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

- $a_{N-1}$: best forward prediction
- $\Gamma_N$: scalar reflection coefficient
- $a_{N-1}^B$: best backward prediction

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

$$\underbrace{R_{N+1} \begin{bmatrix} a_{N-1} \\ 0 \end{bmatrix}}_{\text{I}} \qquad \underbrace{R_{N+1}\, \Gamma_N \begin{bmatrix} 0 \\ a_{N-1}^B \end{bmatrix}}_{\text{II}}$$

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

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

$$r_N^B = [r(N)\ r(N-1) \dots r(1)]^T$$

II. $$R_{N+1}\begin{bmatrix} 0 \\ a_{N-1}^{B} \end{bmatrix} = \begin{bmatrix} \Delta_{N-1} \\ \underline{0} \\ \hat{e}_{N-1} \end{bmatrix}$$

$\Gamma_N$ must satisfy

$$R_{N+1}\, a_N = \begin{bmatrix} \hat{e}_N \\ \underline{0} \end{bmatrix} = \begin{bmatrix} \hat{e}_{N-1} \\ \underline{0} \\ \Delta_{N-1} \end{bmatrix} + \Gamma_N \begin{bmatrix} \Delta_{N-1} \\ \underline{0} \\ \hat{e}_{N-1} \end{bmatrix}$$

where N is the iteration no.

$$\hat{e}_N = \hat{e}_{N-1} + \Gamma_N \Delta_{N-1}$$
$$0 = \Delta_{N-1} + \Gamma_N \hat{e}_{N-1} \quad \rightarrow \quad \Gamma_N = -\frac{\Delta_{N-1}}{\hat{e}_{N-1}}$$

The first recursion is

$$\hat{e}_0 = r(0)$$
$$\Delta_0 = r(1)$$

Recall that for one-step-forward prediction $e = r(0) - p^T \hat{h}$ (from last lecture)

$$\Delta_0 = \sum_{l=0}^{0} (a_0)_l \, r(N-l) \;\downarrow\; 1 = r(1)$$

For $n = 1, \cdots, N$

$$\Gamma_n = -\frac{\Delta_{n-1}}{\hat{e}_{n-1}} \qquad \hat{e}_n = \hat{e}_{n-1} + \Gamma_n \Delta_{n-1}$$
$$\Delta_{n-1} = (r_n^B)^T a_{n-1}$$

$$a_n = \begin{bmatrix} a_{n-1} \\ 0 \end{bmatrix} + \Gamma_n \begin{bmatrix} 0 \\ (a_{n-1})^B \end{bmatrix}$$

Note: Levinson-Durbin algorithm is faster than Schur decomposition and Cholesky decomposition. However, it is more sensitive to computational inaccuracies.

In practice $R$ and $p$ are estimated from data and thus are functions of $n$.

Let $u_n = \begin{bmatrix} x[n] \\ x[n-1] \\ \vdots \\ x[n-(N-1)] \end{bmatrix}$ length $N$.

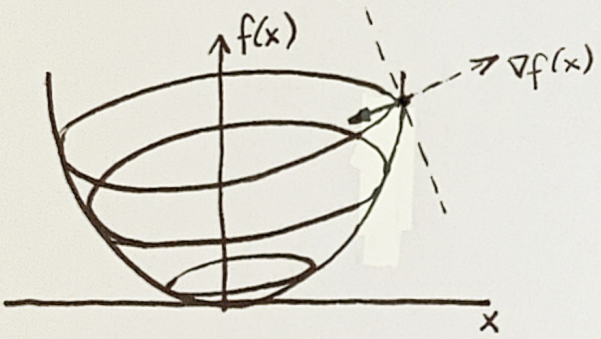
- Candidate filter taps at time $n$: $h_n[0], h_n[1], \ldots, h_n[N-1]$
- Estimate of desired signal

$$\hat{d}_n = \sum_{k=0}^{N-1} h_n[k]\, x[n-k] = h_n^T u_n$$

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

- The filter is adaptive in that $h_n \to h$ that is close to the optimal $h$ as we see more data.
- Can we numerically solve $h_n$? (instead of explicitly computing this function all the time)

## Steepest Descent (Gradient Descent)

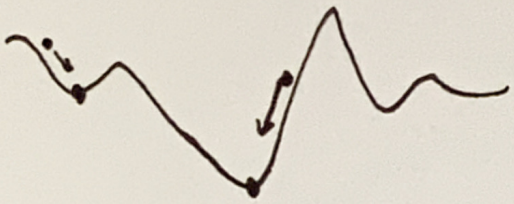



Gradient of an $n$-dimensional function points in the direction of the most rapid change.

Local solution: $h_{n+1} = h_n - \frac{1}{2}\mu \nabla_{h_n} J(n)$

Step size; Gradient of MSE $E[e^2[n]]$ with respect to $h_n$.

Global minimum is hard.



We need to determine the step size $\mu$ and initial choice of where to start from.

$$J = E[e^2[n]] = \sigma_d^2 - 2h_n^T p + h_n^T R h_n \qquad \text{Scalar}$$

$$\nabla_{h_n} J = -2p + 2Rh_n \qquad N\times 1 \text{ vector}$$

Why?

$$\nabla_{h_n} J = \nabla_{h_n}(\sigma_d^2) - 2\nabla_{h_n}(h_n^T p) + \nabla_{h_n}(h_n^T R h_n)$$

verify!

$$\nabla_{h_n}(h_n^T p) \rightarrow \frac{\partial}{\partial h_n[i]} \sum_{j=0}^{N-1} h_n(j)\, p_{j+1} = p_{i+1} \rightarrow \nabla_{h_n}(h_n^T p) = \begin{bmatrix} p_1 \\ \vdots \\ p_N \end{bmatrix}$$

$$\nabla_{h_n}(h_n^T R h_n) \rightarrow \frac{\partial}{\partial h_n[i]} \sum_{j=0}^{N-1}\sum_{k=0}^{N-1} h_n(j) h_n(k)\, r[j-k] = 2\sum_{k=0}^{N-1} h_n(k) r[i-k]$$

$$\rightarrow \nabla_{h_n}(h_n^T R h_n) = 2Rh_n$$

$$h_{n+1} = h_n - \frac{1}{2}\mu \underbrace{(-2p + 2Rh_n)}_{\nabla_{h_n} J}$$

$$= h_n + \mu(p - Rh_n)$$

If $p = Rh_n$ then Wiener-Hopf equation is satisfied.

* Once we hit the minimum we stay there.

We want the error

$$\| h_n - \tilde{h} \| \longrightarrow 0$$

($h_n$: estimate, $\tilde{h}$: ideal)

(the convergence is related to the step size $\mu$)

We need to look at the eigenvalues of the matrix $R = \{\lambda_1, \ldots, \lambda_N\}$

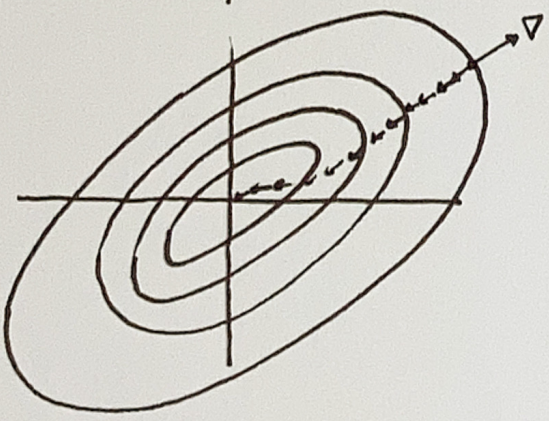
**Theorem** Error $\to 0$ if $|1 - \mu\lambda_k| < 1$ for all $\lambda_k$.

This condition is satisfied if

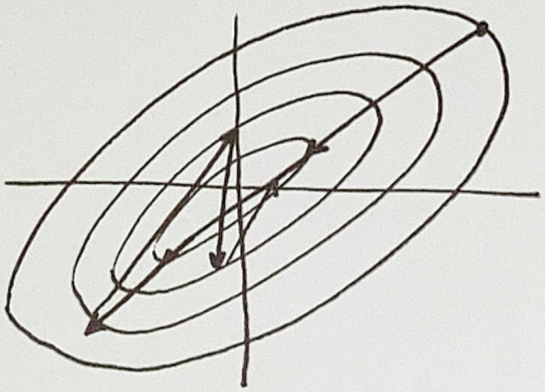
$$0 < \mu < \frac{2}{\lambda_{max}}$$ where $\lambda_{max}$ is the largest eigenvalue of $R$.

## Typical Convergence Problems

Step size too small (overdamped)



Step size too big (underdamped)



* The step size can be adopted as we go.

When we do not know $R$ exactly, we can estimate $R$ as

$$\hat{R}_n = u_n u_n^T$$

using a stochastic gradient algorithm called the Least Mean Square (LMS) that finds the filter weights to minimize a cost (error) function.

## The Least Mean Square (LMS) Algorithm

We do not know $R$ exactly:

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

where $N$ is the filter length

$N \times 1$ vector

$$u_n u_n^T = \begin{bmatrix} x[n]x[n] & x[n]x[n-1] & \cdots & x[n]x[n-(N-1)] \\ x[n]x[n-1] & x[n-1]x[n-1] & \cdots & x[n-1]x[n-(N-1)] \\ \vdots & & & \vdots \\ x[n]x[n-(N-1)] & x[n-1]x[n-(N-1)] & \cdots & x[n-(N-1)]x[n-(N-1)] \end{bmatrix}$$

$$h_n = \begin{bmatrix} h_0[n] \\ h_1[n] \\ \vdots \\ h_{N-1}[n] \end{bmatrix}$$

at time $n$

$$e[n] = d[n] - h_n^T u_n$$

- LMS is based on gradient descent.

- Gradient $> 0$ implies the error would keep increasing positively if the same filter coefficients are used. Therefore, we need to decrement the filter weights.

$J[n] = E[e^2[n]]$ where $e[n]$ is error of current sample.

$$\nabla_{\hat{h}} J[n] = \nabla_{\hat{h}} E[(d[n] - \hat{h}^T u_n)^2]$$

$$\nabla_{\hat{h}} e[n] = \nabla_{\hat{h}} (d[n] - \hat{h}^T u_n) = -u_n \qquad (N \times 1 \text{ vector})$$

$$\nabla_{\hat{h}} J[n] = \nabla_{\hat{h}} E[e[n] \underbrace{(d[n] - \hat{h}^T u_n)}_{e[n]}]$$

$$= -2E[u_n \underbrace{(d[n] - \hat{h}^T u_n)}_{e[n]}]$$

$$= -2E[e[n] u_n]$$

$$\hat{h}_{n+1} = \hat{h}_n - \frac{\mu}{2} \nabla_{\hat{h}_n} J[n] \qquad \text{are the filter taps at time } n+1$$

$$= \hat{h}_n + \mu E[e[n] u_n] \qquad : N \times 1 \text{ vector}$$

$$= \hat{h}_n + \mu (E[u_n] d[n] - E[u_n u_n^T] \hat{h}_n)$$

recall that $d[n]$ and $\hat{h}_n$ are deterministic.

$$= \hat{h}_n + \mu (\underbrace{\hat{P}_n}_{} - \underbrace{\hat{R}_n}_{} \hat{h}_n)$$

$$= \hat{h}_n + \mu (\hat{P}_n - \hat{R}_n \hat{h}_n)$$

$\hat{h}_n$ converges to the Wiener filter solution.

* This is true when the system is LTI and noise process is also stationary.

For the step size $\mu$, we require

$$0 < \mu_n < \frac{2}{tr(R_n)} \qquad tr(R_n) = \sum_{i=0}^{n-1} r(i)$$

$$R_n = \begin{bmatrix} R_n(1,1) & R_n(1,2) & \cdots & R_n(1,n) \\ R_n(2,1) & R_n(2,2) & \cdots & R_n(2,n) \\ & & & \\ R_n(n,1) & R_n(n,2) & \cdots & R_n(n,n) \end{bmatrix}$$

## Least Square (LS) Algorithm

$$\min_x \|Ax - b\|$$

Desired output sequence $d[n]$

Input sequence $x[n]$

Filter $h[n]$

$$d[n] = \overbrace{\sum_{k=0}^{N-1} h[k]x[n-k]}^{\text{actual output}} + e[n]$$

**Problem**: Given $\{x[n]\}$ and $\{d[n]\}$ determine $\{h[0], \ldots, h[N-1]\}$ that minimizes $\sum_{i=n_1}^{n_2} e[i]^2$,

$n_1 = 0$, $n_2 = L-1$ where $L > N$

$$\begin{bmatrix} e[0] \\ \vdots \\ e[L-1] \end{bmatrix} = \begin{bmatrix} d[0] \\ \vdots \\ d[L-1] \end{bmatrix} - \begin{bmatrix} x[0] & x[-1] & x[-2] & \cdots & x[-N+1] \\ x[1] & x[0] & x[-1] & \cdots & x[-N+2] \\ \vdots & & & & \vdots \\ x[L-1] & x[L-2] & \cdots & & x[-N+L] \end{bmatrix} \begin{bmatrix} h[0] \\ h[1] \\ \vdots \\ h[N-1] \end{bmatrix}$$

$L \times 1$ $\quad$ $L \times 1$ $\quad$ $L \times N$ matrix $A$ $\quad$ $N \times 1$

$e = d - Ah$

$e = \begin{bmatrix} e[0] \\ \vdots \\ e[L-1] \end{bmatrix} \rightarrow \|e\|_2^2 = e^2[0] + e^2[1] + \cdots$

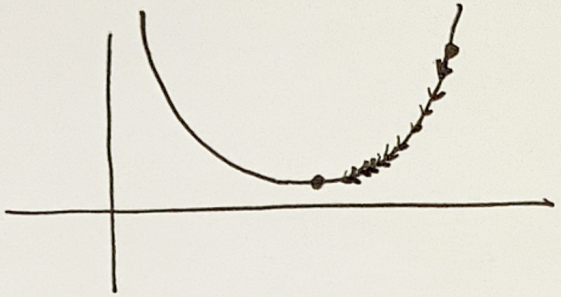
minimize $\|e\|_2^2 = \|d - Ah\|_2^2$

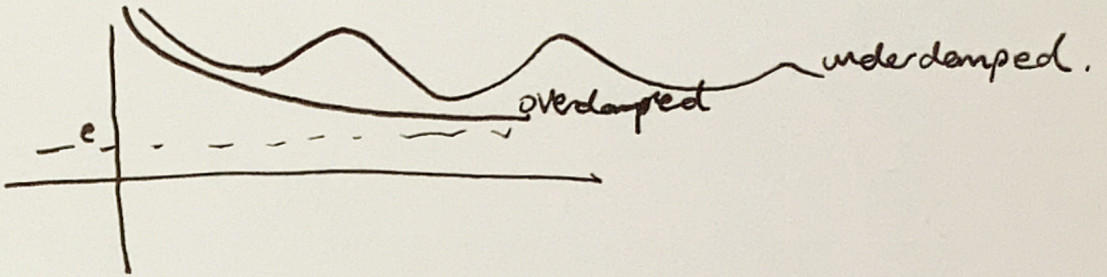
$= (d-Ah)^T (d-Ah)$

$= (d^T - h^T A^T)(d - Ah)$

$= d^T d - d^T Ah - h^T A^T d + h^T A^T A h$

Gradient $= \underline{0} \Rightarrow \frac{\partial}{\partial h[j]} = 0$






$$R_N = \begin{bmatrix} r(0) & r(1) & \cdots & r(N-1) \\ r(1) & r(0) & & r(N-2) \\ & & \ddots & \\ r(N-1) & r(N-2) & & r(0) \end{bmatrix} \qquad R_{N+1} = \begin{bmatrix} r(0) & r(1) & \cdots & r(N-1) & r(N) \\ r(1) & r(0) & \cdots & r(N-2) & r(N-1) \\ & & \ddots & & \\ r(N-1) & r(N-2) & \cdots & r(0) & r(1) \\ r(N) & r(N-1) & & r(1) & r(0) \end{bmatrix}$$

Letting $r_N^B = \begin{bmatrix} r(N) & r(N-1) & \cdots & r(1) \end{bmatrix}^T$

$$R_{N+1} = \begin{bmatrix} R_N & r_N^B \\ (r_N^B)^T & r(0) \end{bmatrix}$$