Today's Lecture

* Least Squares (LS)
* Recursive Least Squares (RLS)

Announcements

Homework 6, due Monday 14^{th}

Final, Wednesday 16^{th} 11:30 - 2:30 pm

Review session, Monday 14^{th}, 10:00 - 11:30 am

Least Squares

$$\min_{x} \|Ax - b\|$$

In our setup, we want to determine $\{h[0], \dots h[N-1]\}$ that minimizes $\sum_{i=0}^{L-1} e[i]^2$ where $L \geq N$. $\Rightarrow d[n] = \sum_{k=0}^{N-1} h[k]x[n-k] + e[n]$

$$\underset{L\times 1}{\begin{bmatrix} e[0] \\ \vdots \\ e[L-1] \end{bmatrix}} = \underset{L\times 1\,(b)}{\begin{bmatrix} d[0] \\ \vdots \\ d[L-1] \end{bmatrix}} - \underset{L\times N\ (A)}{\begin{bmatrix} x[0] & x[-1] & \cdots & x[-N+1] \\ \vdots & & & \vdots \\ x[L-1] & x[L-2] & \cdots & x[-N+L] \end{bmatrix}} \underset{N\times 1\,(x)}{\begin{bmatrix} h[0] \\ \vdots \\ h[N-1] \end{bmatrix}}$$

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

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

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

Gradient $= \underline{0} \implies \frac{\partial}{\partial h[j]} \|e\|_2^2 = 0$

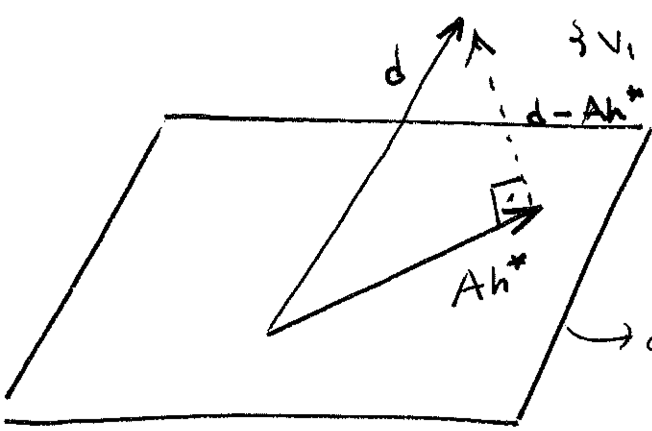
$$\nabla_h \|e\|_2^2 = -A^T d - A^T d + 2A^T A h = \underline{0}$$

$$A^T A h = A^T d \qquad\qquad A : L \times N \quad (L > N)$$

$$\implies h^* = \underbrace{(A^T A)^{-1} A^T}_{\text{pseudo inverse of } A.} d$$

Column space of A

$A = [\underline{v}_1 \ldots \underline{v}_N] \longrightarrow$ column space is the vector space generated by linear combinations of the column vectors $\{v_1, \ldots, v_N\}$



$$Ah^* = [\underline{v}_1 \cdots \underline{v}_N] h^*$$

$\hookrightarrow$ column space of A.

Error is orthogonal to the column space of A.

$$Ah^* \perp (d - Ah^*) \implies A^T(d - Ah^*) = 0$$

$$d = \underbrace{Ah^*}_{\substack{\text{projection to}\\ \text{column space}\\ \text{of A.}}} + \underbrace{(d - Ah^*)}_{\text{error}}$$

$$A^TA = \underbrace{\begin{bmatrix} x[0] & x[1] & \cdots & x[L-1] \\ x[-1] & x[0] & \cdots & x[L-2] \\ \vdots & & & \\ x[-(N-1)] & x[-(N-2)] & \cdots & x[L-N] \end{bmatrix}}_{N\times L} \underbrace{\begin{bmatrix} x[0] & x[-1] & \cdots & x[-(N-1)] \\ x[1] & x[0] & \cdots & x[-N] \\ & & & \\ x[L-1] & x[L-2] & \cdots & x[L-N] \end{bmatrix}}_{L\times N}$$

$$= \begin{bmatrix} x[0]^2 + x[1]^2 + \cdots + x[L-1]^2 & \cdots \\ x[-1]x[0] + x[0]x[1] + x[1]x[2] + \cdots + x[L-2]x[L-1] & \cdots \\ \vdots & \ddots \end{bmatrix}$$

$$\overset{WSS}{\approx} LR$$

autocorrelation

crosscorrelation

$$(A^TA)h^* = A^Td \implies LR\,h^* = A^Td \approx Lp$$

$$R\,h^* \approx p \implies h^* \approx R^{-1}p$$

Wiener-Hopf.

* This is a deterministic approximation to the Wiener filter (if the process is WSS)

For LS, $h^* = (A^T A)^{-1} A^T d$

We want to get $h_{n+1}^* = (A_{n+1}^T A_{n+1})^{-1} A_{n+1}^T d_{n+1}$

from $h_n^* = (A_n^T A_n)^{-1} A_n^T d_n$

We can obtain h_{n+1}^* from h_n^* using <u>Recursive Least Squares (RLS)</u>

$$\min \sum_{i=0}^{n-1} \lambda^{(n-1)-i} e^2[i] = C(h_n)$$

$$0 < \lambda < 1 \qquad \text{(forgetting factors)}$$

$$= 1 . e^2[n-1] + \lambda e^2[n-2] + \lambda^2 e^2[n-3] + \dots$$

$$\frac{\partial C(h_{n-1})}{\partial h_{n-1}(k)} = \sum_{i=0}^{n-1} 2 \lambda^{(n-1-i)} e(i) \frac{\partial e(i)}{\partial h_{n-1}(k)}$$

$$e(i) = d(i) - \sum_{\ell=0}^{N-1} h_{n-1}(\ell) x[i-\ell], \quad i = 0, \dots, n-1$$

$$\frac{\partial e[i]}{\partial h_{n-1}[k]} = -x[i-k]$$

$$\frac{\partial C}{\partial h_{n-1}(k)} = -\sum_{i=0}^{n-1} 2 \lambda^{(n-1-i)} \left[d(i) - \sum_{l=0}^{N-1} h_{n-1}(\ell) x[i-\ell] \right] x[i-k]$$

$$= 0$$

$$\Rightarrow \quad \sum_{i=0}^{n-1} \lambda^{(n-1-i)} d[i]\, x[i-k] = \sum_{l=0}^{N-1} h_{n-1}(l) \left[\sum_{i=0}^{n-1} \lambda^{(n-1-i)} x[i-l]\, x[i-k] \right]$$

Let

$$\left.\begin{array}{lll} \phi_{n-1} = \sum_{i=0}^{n-1} \lambda^{(n-1-i)} u_i u_i^T & & \text{autocorrelation} \\ z_{n-1} = \sum_{i=0}^{n-1} \lambda^{(n-1-i)} u_i\, d[i] & & \text{crosscorrelation} \end{array}\right\} \text{same as before but with } \lambda \text{ weightings.}$$

$$\Rightarrow \quad \underset{\substack{N \times N \\ \text{matrix}}}{\phi_{n-1}}\ \underset{\substack{N \times 1 \\ \text{vector}}}{h_{n-1}} = \underset{\substack{N \times 1 \\ \text{vector}}}{z_{n-1}} \quad \text{at time } n-1 \qquad (R h = p)$$

h^*_{n-1} is the optimal solution at time $n-1$.

$$\phi_n h_n^* = z_n \quad \Rightarrow \quad h_n^* = \phi_n^{-1} z_n$$

where

$$\phi_n = \sum_{i=0}^{n} \lambda^{n-i} u_i u_i^T$$

$$z_n = \sum_{i=0}^{n} \lambda^{n-i} u_i\, d[i]$$

$$\phi_n = \underbrace{\sum_{i=0}^{n-1} \lambda^{n-i} u_i u_i^T}_{\lambda \phi_{n-1}} + u_n u_n^T = \lambda \phi_{n-1} + u_n u_n^T$$

$$z_n = \lambda z_{n-1} + u_n d[n]$$

Matrix Inversion Lemma (MIL)

A, B positive definite $M \times M$

C $M \times N$

D positive definite $N \times N$

If $A = B^{-1} + C D^{-1} C^T$

Then $A^{-1} = B - BC(D + C^T B C)^{-1} C^T B$

Let $A = \phi_n$ $\qquad \overbrace{\phi_n}^{A} = \overbrace{\lambda \phi_{n-1}}^{B^{-1}} + \overbrace{u_n}^{C}\overbrace{u_n^T}^{C^T}$

$B^{-1} = \lambda \phi_{n-1}$

$C = u_n$

$D^{-1} = 1$

Using MIL,

$$\phi_n^{-1} = \lambda^{-1}\phi_{n-1}^{-1} - \lambda^{-1}\phi_{n-1}^{-1} u_n (1 + u_n^T \lambda^{-1} \phi_{n-1}^{-1} u_n)^{-1} u_n^T \lambda^{-1} \phi_{n-1}^{-1}$$

$P_n = \phi_n^{-1}$

$$P_n = \frac{1}{\lambda} P_{n-1} - \frac{1}{\lambda^2} \frac{P_{n-1} u_n u_n^T P_{n-1}}{1 + \frac{1}{\lambda} u_n^T P_{n-1} u_n}$$

$$= \frac{1}{\lambda} P_{n-1} - \frac{1}{\lambda} k_n u_n^T P_{n-1}$$

where $k_n = \dfrac{1}{\lambda} \dfrac{P_{n-1} u_n}{1 + \frac{1}{\lambda} u_n^T P_{n-1} u_n}$ is the gain vector.

$$k_n = \left(\frac{1}{\lambda} P_{n-1} - \frac{1}{\lambda} k_n u_n^T P_{n-1}\right) u_n = P_n u_n \qquad (\text{try to show this})$$

$$\Rightarrow \quad h_n^* = \phi_n^{-1} z_n = P_n z_n$$

$$= P_n (\lambda z_{n-1} + u_n d[n])$$

$$= \lambda P_n z_{n-1} + P_n u_n d[n]$$

$$= \lambda \left(\frac{1}{\lambda} P_{n-1} - \frac{1}{\lambda} k_n u_n^T P_{n-1}\right) z_{n-1} + P_n u_n d[n]$$

$$= P_{n-1} z_{n-1} - k_n u_n^T P_{n-1} z_{n-1} + P_n u_n d[n]$$

$$= h_{n-1}^* - k_n u_n^T h_{n-1}^* + k_n d[n]$$

$$= h_{n-1}^* - k_n (u_n^T h_{n-1}^* - d[n])$$

$$= h_{n-1}^* + k_n \xi_n \quad \text{where} \quad \xi_n = d[n] - u_n^T h_{n-1}^*$$

Recursive Least Squares (RLS)

1. Gain vector $$k_n = \frac{\frac{1}{\lambda} P_{n-1} u_n}{1 + \frac{1}{\lambda} u_n^T P_{n-1} u_n}$$

2. Error from previous estimate

$$\xi_n = d[n] - h_{n-1}^{*T} u_n = d[n] - u_n^T h_{n-1}^*$$

3. Update filter

$$h_n^* = h_{n-1}^* + k_n \xi_n$$

4. Update $\phi_n^{-1} = P_n$ where $P_n = \frac{1}{\lambda} P_{n-1} - \frac{1}{\lambda} k_n u_n^T P_{n-1}$

In practice, RLS converges much faster than LMS.

It does not depend on eigenvalues of Φ_n.