

Normalized LMS Algorithm

- Recall the *standard LMS algorithm*:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu \mathbf{u}(n) e^*(n)$$

- *Normalized LMS algorithms*:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{1}{\|\mathbf{u}(n)\|^2} \mathbf{u}(n) e^*(n)$$

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{\|\mathbf{u}(n)\|^2} \mathbf{u}(n) e^*(n)$$

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{a + \|\mathbf{u}(n)\|^2} \mathbf{u}(n) e^*(n)$$

- The first normalized LMS algorithm follows from "*minimal disturbance*":

$$\begin{aligned} & \min_{\hat{\mathbf{w}}(n+1)} \|\hat{\mathbf{w}}(n+1) - \hat{\mathbf{w}}(n)\|^2 \\ & \text{subject to } \hat{\mathbf{w}}^H(n+1) \mathbf{u}(n) = d(n) \end{aligned}$$

Block Adaptive Filters

- Recall the definitions:

$$\mathbf{u}(n) = [u(n), u(n-1), \dots, u(n-M-1)]$$

$$\mathbf{w}(n) = [w_0(n), w_1(n), \dots, w_{M-1}(n)]$$

- *Block data processing:*

$$y(kL-i) = \hat{\mathbf{w}}^H(k) \mathbf{u}(kL-i); \quad i = 0, 1, \dots, L-1$$

- *Block LMS Algorithm:*

$$\hat{\mathbf{w}}(k+1) = \hat{\mathbf{w}}(k) + \mu \sum_{i=0}^{L-1} \mathbf{u}(kL-i) e^*(kL-i)$$

- Note: $-\frac{2}{L} \sum_{i=0}^{L-1} \mathbf{u}(kL-i) e^*(kL-i)$ is the "averaged estimate" of the gradient $-2E(\mathbf{u}(n)e^*(n))$.

- The choice of step size (from an approximate analysis):

$$0 < \mu < \frac{2}{L \times \text{tr}(\mathbf{R})} \quad (\text{Normally, } L = M)$$

Fast LMS Algorithm

- For each k , use "overlap-and-save FFT" to compute (assuming real data)

$$y(kM - i) = \hat{\mathbf{w}}^T(k) \mathbf{u}(kM - i); \quad i = 0, 1, \dots, M - 1$$

Define:

$$\hat{\mathbf{w}}_a(k) = \begin{bmatrix} \hat{\mathbf{w}}(k) \\ \mathbf{0} \end{bmatrix}; \quad 2M \times 1$$

$$\mathbf{u}'_a(k) = \begin{bmatrix} \mathbf{u}'(k-1) \\ \mathbf{u}'(k) \end{bmatrix}; \quad 2M \times 1$$

$$\text{where } \mathbf{u}'(k) = [u(kM - M + 1) \quad \dots \quad u(kM)]^T$$

$$\mathbf{y}'(k) = [y(kM - M + 1) \quad \dots \quad y(kM)]^T$$

Then:

$$\mathbf{y}'(k) = \text{last } M \text{ elements of } (\hat{\mathbf{w}}_a(k) \otimes_{2M} \mathbf{u}'_a(k))$$

where

\otimes_{2M} denotes *circular convolution of two 2M-element vectors*.

- For each k , use "overlap-and-save FFT" to compute the vector:

$$\phi(k) \triangleq \sum_{i=0}^{M-1} \mathbf{u}(kM - i)e(kM - i);$$

Define

$$\mathbf{u}_a(k)^R = \text{circularly_reversed_vector_of_}\mathbf{u}'_a(k)$$

$$\mathbf{e}'_a(k) = \begin{bmatrix} \mathbf{0} \\ \mathbf{e}'(k) \end{bmatrix}$$

$$\text{where } \mathbf{e}'(k) = [e(kM - M + 1) \ \cdots \ e(kM)]^T$$

Then

$$\phi(k) = \text{first_}M\text{_elements_of_}\left(\mathbf{u}_a(k)^R \otimes_{2M} \mathbf{e}'_a(k)\right)$$

- Note that:

$$\begin{aligned} & \mathbf{u}'_a(k)^R \otimes_{2M} \mathbf{e}'_a(k) \\ &= \text{IFFT}\left(\text{FFT}\left(\mathbf{u}'_a(k)^R\right) \bullet \text{FFT}\left(\mathbf{e}'_a(k)\right)\right) \\ &= \text{IFFT}\left(\text{FFT}\left(\mathbf{u}'_a(k)\right)^* \bullet \text{FFT}\left(\mathbf{e}'_a(k)\right)\right) \end{aligned}$$

- How much is the computational saving?

Frequency-Domain Normalization

- Define

$$\mathbf{v}_a(k) \triangleq FFT(\mathbf{u}_a(k))$$

$$\mathbf{v}_a(k) = \begin{bmatrix} v_{a,0}(k) \\ \vdots \\ v_{a,2M-1}(k) \end{bmatrix}$$

where each element corresponds to a *frequency bin*.

- To track the power in the *i-th* frequency bin:

$$P_i(k) = \gamma P_i(k-1) + (1-\gamma) |v_{a,i}(k)|^2; \quad i = 0, 1, \dots, 2M-1$$

- To ensure (approximately) equal rate of convergence for all frequency components (i.e., *modes*):

$$v_{a,i}(k) \leftarrow \frac{v_{a,i}(k)}{P_i(k)}$$

where " \leftarrow " here means "replaced by".