

انتخاب طول Block در LMS

در 1981 Clark et al. پیشنهاد کردند که  $L = M$  انتخاب شود.

اگر  $L > M$  : مابسات اضافه انجام می شود، چون تخمین بردار گراوین (محدود نمونه) اطلاعاتی بیشتر از خود نمونه استاندارد می کند.  
 اگر  $L < M$  : بعضی از tap weights تلف می شوند. (به نظر من صحیح نیست).

به نظر من:

$L > M$  : مابسات اضافه. دلیل ندارد بیشتر صبر کنیم. همه ضرایب ورودی خود را بشماریم و داده آمد.

$L < M$  : اطلاعات کافی را  $\hat{w}$  نمی بینیم. جمع چند قطعات.  
 به نظر من باید دلیل این را با استفاده از FFT است.

Fast LMS Algorithm یا FBLMS

در آنگونه که Block LMS در کانولوشن وجود دارد.

① کانولوشن قطب مناسب فریبی از روی ورودی:  $y(kL+i) = \hat{w}^T(k) u(kL+i) = \sum_{q=0}^{M-1} \hat{w}_q(k) u(kL+i-q)$   
 (مستند 45 جعقی)

② در مابست همگی  $u$  و  $e$  که در واقع نوعی کانولوشن برعکس است (reverse conv):

$\phi(k) = \sum_{i=0}^{L-1} u(kL+i) e(kL+i) \Rightarrow \phi_z(k) = \sum_{i=0}^{L-1} u(kL+i-z) e(kL+i)$   
 (z = 0, 1, ..., L-1)

میاده سازی این کانولوشن با استفاده از DFT و روش مابست سریع کانولوشن با استفاده از FFT و روش

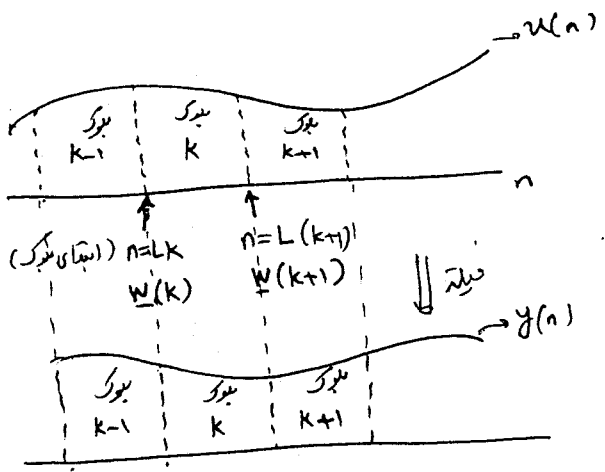
Fast LMS algorithm ← overlap-save (Clark et al, 1980; Ferrara, 1980)

← overlap-add (در اینجا) ولی در حالت overlap 50% ← overlap-save

مراحل تبدیل BLMS به FBLMS :

در قدم اول BLMS را در حوزه فرکانس پیاده سازی می کنیم. به این ترتیب که کانولوشن های همپوشانی آنرا با استفاده از Block Convolution و الگوریتم overlap-save پیاده سازی می کنیم.

BLMS :



در انتهای هر بلوک  $w(k)$  update می شود.  
 در کل بلوک  $y$  از صفر کردن  $u$  با همان  $w$  آفریبت می آید.  
 $KL \leq n \leq KL+L-1$   
 برای کل بلوک  $k$  ام داریم :

$$y(n) = \hat{w}(k)^H u(n) \quad KL \leq n \leq KL+L-1$$

$$y(KL+i) = \hat{w}(k)^H u(KL+i), \quad 0 \leq i \leq L-1$$

توجه : ← برای حالت نمونه های ابتدایی  $y$  در هر بلوک (مثلاً  $i=0$ ) به مقدار  $u$  قبل از این بلوک احتیاج داریم. هیچ تقریبی زده نمی شود و از همان مقدار واقعی استفاده می شود. به عبارتی نباید به اشتباه تصور شود.

که مقدار  $u$  در بلوک  $k$  ام برای تعیین خروجی بلوک  $k$  ام کافی است. برای  $L=M$ ، ما تقریباً به تمام بلوک  $(k-1)$  ام ورودی نیاز داریم (به عبارت دقیقتر به  $M-1$  نمونه آخر بلوک  $(k-1)$  ام ورودی هم نیاز داریم) ← این نکته باید در پیاده سازی Block Convolution مورد توجه قرار گیرد. اما جالب است که اگر بلوک های Block Convolution را به طول  $2M$  ( $L=M$ ) در نظر بگیریم و از  $50\%$  overlap استفاده کنیم همه چیز به طور خودکار درست می شود.

دورانی  $(2M)$  نقطه ای

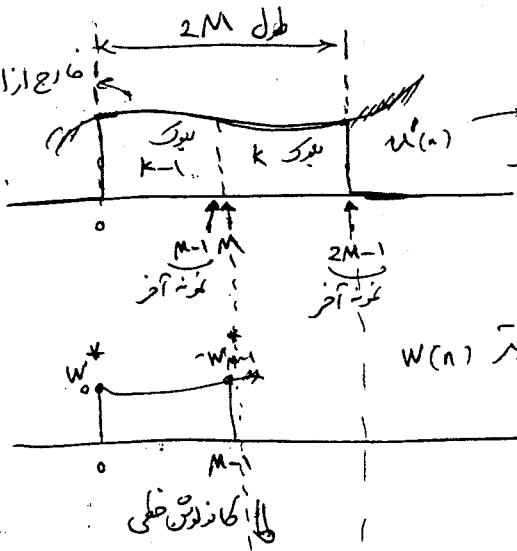
← یعنی بلوک  $k$  ام در  $(k-1)$  ام ورودی را کانولوشن می کند با  $w$  که در این نقطه اول را در دوری داریم.

آفریبت می آید، همان  $y(n)$  در بلوک  $k$  ام است. دلیل :

← ابتدا کانولوشن معمولی (اصولی) اینها را در نظر بگیرید :

۳

فایده از این موردده ضروری است



دستور  $u(n)$  که خارج از این دو بلوک هنر شده است

کانولوشن خطی این بلوک از دستاورد به طول  $M+2M-1=3M-1$

نکته مهم: دقت شود که بلوک  $M$  تایی

اول خروجی  $y(n)$ ، خروجی صحیح نیست، چون  $u(n)$  نمی هستند (مربوط به بلوک  $k-2$ )

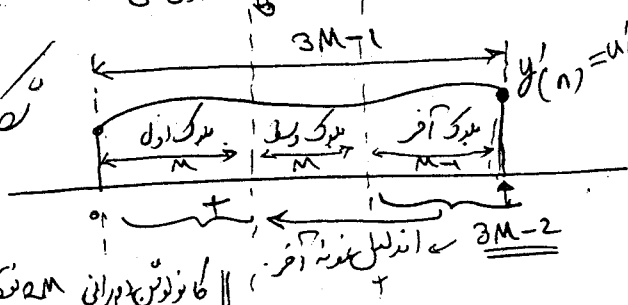
که صفر فرض شده اند. به همین دلیل نیز بلوک آخر خروجی (طول  $M-1$ ) نیز صحیح نیست

برای خروجی صحیح نیست

(اعداد آن چرت و پرت است).

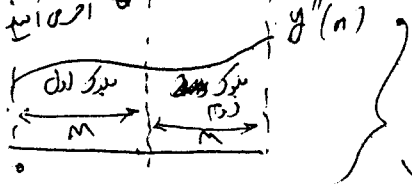
تنها بلوک وسط، خروجی صحیح است که همان بلوک  $k$  ام خروجی است که به دنبال آن هستیم.

شکل (a)



کانولوشن دوران  $2M$  نقطه ای. سینال  $2M$  تا  $2M$  تا دم روشن پیچیده می شود.  $M-1$  نمونه آخر است و بلوک اول را کاملاً فریب می کشد.  $M-1$  نمونه آخر را فریب می کشد.

شکل (b)



با انجام کانولوشن دوران  $M-1$  نمونه آخر (بلوک آخر) شکل (b)،  $M-1$  نمونه اول (تقریباً کل بلوک اول) جمع می شوند. اما بلوک وسط (بلوک دوم) که همان میزبان است که ما به دنبال آن هستیم دست نخورده باقی می ماند.

برای مابقی بلوک  $k$  ام خروجی می توان سینال بلوک  $(k-1)$  ام  $(k)$  ام  $(2M)$  نقطه ای را با فیلتر  $(M)$  نقطه ای کانولوشن دورانی کرد (مثلاً با DFT) و پس نصف اول را دور انداخت.

نکته: همانطور که از شکل فوق بدین است، اگر بجای DFT ای  $2M$  نقطه ای، DFT ای  $M-1$  نقطه ای می گیریم و بجای کل بلوک  $(k-1)$  ام از  $M-1$  نمونه آخر  $(k)$  ام استفاده می کنیم، باز هم میزبان دست نخورده در واقع در فرهنگ بردی همین کار را کرده است (ما نمی کنیم!)

$$u(k) = \begin{bmatrix} u_A(k) \\ u_B(k) \end{bmatrix} \quad u(k) = \text{FFT} \{ u_A(k) \} \quad u(k) = \text{FFT} \{ u_B(k) \}$$

بنابراین پیاده سازی سریع کارتون زیر (مابین بلوک k ام فرعی):

$$y(kL+i) = \hat{W}^H(k) \underline{u}(kL+i) \quad (i \leq L-1)$$

$\underline{u}'(k) = \begin{bmatrix} u_B(k-1) \\ \vdots \\ u_B(k) \end{bmatrix}$  (از این نقطه)  
 $L=M$   
 $\underline{u}'(k) = \begin{bmatrix} u(kL-L) \\ u(kL-L+1) \\ \vdots \\ u(kL-1) \\ u(kL) \\ u(kL+1) \\ \vdots \\ u(kL+L-1) \end{bmatrix}$  (بلوک ام (k-1) و بلوک ام k)  
 صعودت زیر می شود:  $(L=M)$   
 $\underline{U}(k) = \text{FFT} \{ \underline{u}'(k) \}$  (از این نقطه)

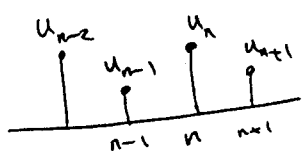
$\underline{W}(k) = \text{FFT} \left\{ \begin{bmatrix} \underline{w}^*(k) \\ \vdots \\ \underline{0}_{M \times 1} \end{bmatrix} \right\}$   
 این \* در کتا Haykin و فیلتر نیست چون در هر دو آن کدریم را با هم دیتای عیبی نسبت در دهانند

$\underline{y}_B(k) = \text{The last } M \text{ elements of } \text{IFFT} \{ \underline{U}(k) \odot \underline{W}(k) \}$   
 که ضرب عنصر به عنصر (از این نقطه)  $(L=M)$

توجه مهم: منظور از (نقطه) این است که در کنار بردارای فوق نوشته شده است و

چون می خواهیم بردار را ضرب نقطه به نقطه بکنیم باید آنها را در یک جهت تشکیل دهیم

$\underline{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{M-1} \end{bmatrix}$  (توی کده دریم) (از این نقطه)



وقتی  $u(n)$  را در ابتدای دریم تعریف کردیم عناصر آنرا برعکس زمان

$\underline{u}(n) = \begin{bmatrix} u(n) \\ u(n-1) \\ \vdots \\ u(n-M+1) \end{bmatrix}$  (توی کده دریم) (از این نقطه)

اینکار برای کارتون زیری خوب بود و با یک ضرب انجام می شد:  $\underline{w}^H \cdot \underline{u}(n)$

اما حالا می خواهیم در حوزه فرکانسی نمونه را با هم ترتیب در هم ضرب کنیم. پس بلوک  $\underline{u}_B(k)$  را (چه در ورودی)

و چه در خروجی) ضعیف تعریف می کنیم:

$\underline{u}_B(k) = \begin{bmatrix} u(kL) \\ u(kL+1) \\ \vdots \\ u(kL+L-1) \end{bmatrix}$  (از این نقطه)  
 بلوک ام (k-1) و بلوک ام k  
 $n=kL$  و  $n=kL+L-1$  (از این نقطه)



نسخه دوم FBLMS (اصلاح نسخه اول، هنوز نسخه نهایی نیست):

در نسخه‌ای که بهت آوردم، ملاحظه می‌شود که تنها جایی که فیلتر استفاده شد در step 1 است (عبارت فیلتر).  
 در آن استفاده شد. پس لازم نیست که فیلتر را در حوضه زمان حساب کنیم (step 4) و بجای آن مستقیماً  
 فیلتر را در حوضه فرکانس update می‌کنیم. یعنی step 4 را بصورت زیر تغییر می‌دهیم:

از طریق رابطه step 4 نسخه قبل DFT یا 2M نقطه‌ای بدیم  $\Rightarrow \underline{W}(k+1) = \underline{W}(k) + \mu \text{FFT} \left\{ \begin{bmatrix} \Phi(k) \\ \Phi_{M \times 1} \end{bmatrix} \right\}$

پس نسخه جدید (دوم FBLMS) چنین می‌شود:

Step 1: محاسبه برداری  
 $\underline{u}'(k) = \begin{bmatrix} \text{بلاک } k-1 \\ \text{بلاک } k \end{bmatrix} \xrightarrow{\text{از زمان به فرکانس}} = \begin{bmatrix} u(kL-L) \\ \vdots \\ u(kL-1) \\ u(kL) \\ \vdots \\ u(kL+L-1) \end{bmatrix}, \underline{U}(k) = \text{FFT} \{ \underline{u}'(k) \}$   
 $\underline{W}(k) = \text{FFT} \left\{ \begin{bmatrix} \underline{w}(k) \\ \Phi_{M \times 1} \end{bmatrix} \right\}$   
 $\Rightarrow \underline{y}_B(k) \xrightarrow{\text{از فرکانس به زمان}} = \text{Last } M \text{ element of } \text{IFFT} \{ \underline{U}(k) \otimes \underline{W}(k) \}$

Step 2: محاسبه خطا  
 $\underline{e}(k) = \underline{d}_B(k) - \underline{y}_B(k), \underline{E}(k) = \text{FFT} \left\{ \begin{bmatrix} \Phi_{M \times 1} \\ \underline{e}^*(k) \end{bmatrix} \right\}$

Step 3: تعیین ضرایب  
 $\underline{\Phi}(k) = \text{First } M \text{ elements of } \text{IFFT} \{ \underline{U}(k) \otimes \underline{E}(k) \}$   
 $\underline{\Phi}(k) = \text{FFT} \left\{ \begin{bmatrix} \underline{\Phi}(k) \\ \Phi_{M \times 1} \end{bmatrix} \right\}$

Step 4: Filter Update:  
 $\underline{W}(k+1) = \underline{W}(k) + \mu \text{FFT} \left\{ \begin{bmatrix} \Phi(k) \\ \Phi_{M \times 1} \end{bmatrix} \right\} \rightarrow \text{update در حوضه فرکانس}$

این نسخه همانی است که بلوک دیگر در ام آن در شکل 10.2 - Haykin (3rd Ed.) 454 کپی شده است. صفاً این بخش را کم می‌شود.

نسخه سوم FBLMS (اصلاح نسخه دوم) - باز هم اصلاح خواهد شد.

مقدارهای (3) و (4) را ضمن تغییر هم : به قدم (5) بیان می کنیم نگاه کنیم محدودیتی روی گرایان می گذاریم (به این ترتیب که M نمونه آخر IFFT آن باید صفر باشند). می توانیم این محدودیت را روی خودمان نیز بگذاریم.

به این ترتیب step 3, step 4 ضمن یک شوند:

Step 1 → صفر قبل  
Step 2 →

Step 3: Filter Update

$$\underline{W}(k+1) = \underline{W}(k) + \mu \underline{U}^*(k) \odot \underline{E}(k)$$

Step 4: Filter (Tap-Weights) constraint

$$\underline{W}(k+1) = \text{FFT} \left\{ \begin{array}{l} \text{First Elements of IFFT } \underline{W}(k+1) \\ \odot_{M \times 1} \end{array} \right\}$$

نسخه سوم

version 1 (1) → اصلاح اولیست - ver. 0.3

نسخه نهمی FBLMS

آنگونه FBLMS تا اینجا هیچ ضرورتی بجز یک پیاده سازی سریع برای BLMS. یعنی خود به دستگیر شدن آن دقیقاً با BLMS می کنند و هیچ برتری ای (یا اصولاً تفاوتی) از نظر سرعت همگرای دیجیتال فزونی و ... با BLMS ندارد. (بای M سی با MS ، LMS) تفاوت اصلی آن با LMS عادی آنست که حجم محاسبات آن کمتر است.

محاسبه برون حجم محاسبات نسبت به نسخه اولیست ...  
→ مطالب مربوط به حجم محاسبات که در همین بیان شده مال اینجاست.

→ در اینجا می خواهیم که اصلاحی را در آنگونه انجام دهیم که به قیمت افزایش حجم محاسبات، سرعت همگرای آنرا خیلی زیادتر کند.

→ اصلاحی که انجام می دهیم در Step 3 آنگونه بالا (نسخه سوم) است. موقتاً Step 4 را از میان می کشیم، مثل اینست که هر کدام از مراتب DFT (منتهی W(k)) یک LMS جداگانه دارند، پس لازم نیست که M همه آنها یکی باشند. در واقع در هر کدام اگر انرژی سکینال در یکی از باندها کمتر است M مساعده می تواند برای انرژی

۹

سرعت همگرایی بیشتر کند و برعکس. در نتیجه بجای آنکه  $\mu$  هر را مساوی بگیریم، در باند نام  
 (برای ضرب DFT نام) آنرا برابر  $\mu_i = \frac{\mu_0}{P_i}$  میگیریم که در آن  $P_i$  انرژی مؤلفه نام  
 $U(k)$  است  $(P_i = E\{|U_i(k)|^2\})$  و ضریب تخمین زده می شود:

$$P_i(k) = \delta P_i(k-1) + (1-\delta) |U_i(k)|^2, \quad 0 < \delta < 1$$

که عددی نزدیک به 1.

در مثل اینست که در باند نام، بجای LMS از فرقی MNLMS استفاده کرده باشیم.

با این حال طوری که هم در آن نگاه کرد. در LMS معمولی هر ضریب فیلتر در هر گامی از نموداری مختلف  
 تأثیر دارد.  $\mu$  را باید طوری انتخاب کنیم که همه مودها همگرا شوند  $(\frac{2}{\lambda} < \mu < 0.2)$ . پس برای آن مودی که  
 $\lambda$  بزرگتر دارد،  $\mu$  کوچکتر انتخاب کنیم، در نتیجه مودها  $\lambda$  کوچک همگرایی اش کند شود (یعنی  $\mu$  بزرگ)

که eigenvalue spread بالا یا همگس زیاد بین مؤلفه ها، همگرایی LMS را کند می کند. اما در اینجا مجبور  
 نیستیم که  $\mu$  را مساوی بگیریم. در واقع مثل اینست که مودهای مختلف در حوزه فرکانس از هم جدا می شوند  
 و هر ضریب  $W(k)$  تنها مؤثر همگرایی یک مود است. در واقع  $\lambda_i = \frac{S(\omega_i) |Q(\omega_i)|^2}{|Q(\omega_i)|^4}$ ،  $\lambda_i$  تقریباً

انرژی سکنال در باند فرکانسی نام  $\omega_i$  را نشان می دهد (البته اینطوری  $M$  باند میشود  
 نه  $2M$  باند) حرف تقویمی است. و نیز مالزیه کردن  $\mu$  با عکس توان تقریباً مثل اینست که از  
 $\mu_i = \frac{\mu_0}{\lambda_i}$  استفاده کرده باشیم و همه مودها را سریع همگرا کنیم. نگاه دیگر اینست که اینکار همگرا

سکنال در حوزه فرکانسی را از مالزیه می کند بطوریکه  $S(\omega)$  کمینوافت کند، یعنی  $\frac{S_{max}}{S_{min}}$  و در نتیجه  $\frac{\lambda_{max}}{\lambda_{min}}$   
 کم شود (eigenvalue spread کمتر) و در نتیجه سرعت همگرایی بالاتر رود. (یعنی همگرا شدن سریعتر مودها  
 می شوند).

در نتیجه الگوریتم نامی FBLMS طبق ضریب بدست می آید.

نسبت LMS معمولی، این الگوریتم هم همگرایی سریعتر دارد و هم حجم محاسبات کمتر.  
 عیب آن نسبت به LMS اینست که تأثیر در محاسبه فرقی است (به اندازه یک بزرگ تأثیر دارد). یک بزرگ  
 دنیا جمع می شود پس همه فرقی همزمان ترکیب می شود. عیب دوم که فصل مهم نیست پیچیدگی بیشتر در پیاده سازی  
 الگوریتم است.

# UCL FBLMS (التكرار)

L=M

20/3/2019

Inputs: - Tap-weight vector:  $\underline{W}(k)$

- signal power estimates:  $P_i(k)$

- Extended input vector:  $\underline{u}'(k) = \begin{bmatrix} u(kL-h) \\ u(kL-1) \\ u(kL) \\ u(kL+L-1) \end{bmatrix}$   $k=1$   $k \neq 1$

- Extended desired output vector:  $\underline{d}_B(k) = \begin{bmatrix} d(kL) \\ d(kL+1) \\ d(kL+L-1) \end{bmatrix}$

Outputs: - Filter output:  $\underline{y}_B(k) = \begin{bmatrix} y(kL) \\ y(kL+1) \\ \vdots \\ y(kL+L-1) \end{bmatrix}$

Step 1: Filtering (التصفية)

$$\underline{U}(k) = \text{FFT} \{ \underline{u}'(k) \}$$

$$\underline{y}_B(k) = \text{Last } M \text{ elements of } \text{IFFT} \{ \underline{U}(k) \odot \underline{W}(k) \}$$

Step 2: Error Estimation

$$\underline{e}_B(k) = \underline{d}_B(k) - \underline{y}_B(k)$$

Step 3: step normalization

for  $i=0$  to  $2M-1$

$$P_i(k) = \delta P_i(k-1) + (1-\delta) \frac{|U_i(k)|^2}{|U_i(k)|^2}$$

$$M_i(k) = \frac{P_0}{P_i(k)}$$

end

$$\underline{M}(k) = \begin{bmatrix} M_0(k) \\ \vdots \\ M_{2M-1}(k) \end{bmatrix}$$

Step 4: Tap-Weight (Filter) Adaptation

$$\underline{E}(k) = \text{FFT} \left\{ \begin{bmatrix} \underline{0}_{M \times 1} \\ \underline{e}_B^*(k) \end{bmatrix} \right\}$$

$$\underline{W}(k+1) = \underline{W}(k) + \underline{M}(k) \odot \underline{U}(k) \odot \underline{E}(k)$$

Step 5: Tap-weight Constraint

$$\underline{W}(k+1) = \text{FFT} \left\{ \begin{bmatrix} \text{First } M \text{ elements of } \text{IFFT}(\underline{W}(k+1)) \\ \underline{0}_{M \times 1} \end{bmatrix} \right\}$$

