深蓝学院
shenlanxueyuan.com

自适应滤波方法（一）

主讲人 宋 辉

清华大学电子工程系 博士
滴滴AI Labs 语音技术部

3.1 LMS算法

滤波器——改变信号频谱

模拟滤波器：由R、L、C构成的模拟电路，例如A/D前的抗混叠滤波器（anti-alias filter）

数字滤波器：由数字加法器、乘法器、延时器构成，基于数字信号运算实现。

自适应滤波器

自适应滤波器：一种能够根据输入信号自动调整自身参数的数字滤波器。

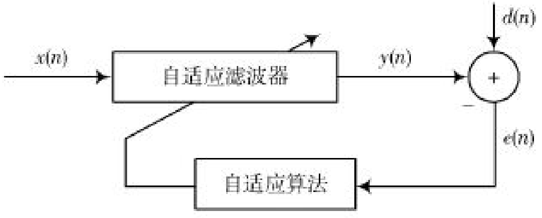
非自适应滤波器：具有静态滤波器系数的数字滤波器，这些静态系数构成了滤波器的传递函数。

3.1.1 自适应滤波器的应用

- 对于一些应用（如系统辨识、预测、噪声消除等），我们无法事先知道需要进行操作的参数，必须使用自适应的系数进行处理，这种情况下通常使用自适应滤波器。

  自适应滤波器处理语音信号时，不需要事先知道输入信号和噪声的统计特性，滤波器自身能够在工作过程中学习或估计信号的统计特性，并以此为依据调整自身参数，以达到某种准则/代价函数下的最优滤波效果。

  一旦信号统计特性发生变化，还可以跟踪这种变化，重新调节参数，使滤波性能重新达到最优。因此，自适应滤波是处理非平稳信号的一种有效手段。



自适应滤波器的一般结构

3.1.2 LMS算法

我们从一个N阶线性系统出发……

我们的目标：

设计一个N阶滤波器，它的参数为 $\mathbf{w}(n)$，则滤波器输出为：

$$y(n) = \sum_{i=0}^{N-1} w_i(n)x(n-i) = \mathbf{w}^T(n)\mathbf{x}(n) = \mathbf{x}^T(n)\mathbf{w}(n) \tag{3.1}$$

上式中：

$$\mathbf{x}(n) = [x(n), \quad x(n-1), \quad \ldots \quad x(n-N+1)]^T \tag{3.2}$$

$$\mathbf{w}(n) = [w_0(n), \quad w_1(n), \quad \ldots \quad w_{N-1}(n)]^T \tag{3.3}$$

期望输出为 $d(n)$，定义误差信号：

$$e(n) = d(n) - y(n) = d(n) - \mathbf{w}^T(n)\mathbf{x}(n) \tag{3.4}$$

3.1.2 基本LMS算法

根据最小均方误差(MMSE)准则，最小化目标函数： $J(\mathbf{w})$

$$J(\mathbf{w}) = E\left\{|e(n)|^2\right\} = E\left\{\left|d(n) - \mathbf{w}^T(n)\mathbf{x}(n)\right|^2\right\} \tag{3.5}$$

为了最小化均方误差函数，需计算 $J(\mathbf{w})$ 对 $\mathbf{w}$ 的导数，令导数为零：

$$E\left\{\mathbf{x}(n)\mathbf{x}^T(n)\right\}\mathbf{w}(n) - E\left\{\mathbf{x}(n)d(n)\right\} = \mathbf{R}\mathbf{w} - \mathbf{r} = \mathbf{0} \tag{3.6}$$

可得到：

$$\boxed{\mathbf{w}_{opt} = \mathbf{R}^{-1}\mathbf{r}} \tag{3.7}$$

(3.7)定义的滤波器称为Wiener滤波器。Wiener滤波器是均方误差最小意义上的统计最优滤波器。

3.1.2 基本LMS算法

由于涉及数学期望，在实际应用中，绝大多数情况下我们无法获得信号真实的自相关矩阵，以及输入信号与期望输出之间的互相关向量，因此，实际应用中，我们用(3.5)式的瞬时梯度，代替数学期望：

$$\hat{\nabla}(n) = -2e(n)\mathbf{x}(n) \tag{3.8}$$

于是，我们得到了标准时域LMS算法的更新公式：

$$\boxed{\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu\mathbf{x}(n)e(n)} \tag{3.9}$$

(3.9)式是逐点更新，也就是说，每当给定一个新的 $x(n)$ 和 $d(n)$，滤波器系数就更新一次。

3.1.2 基本LMS算法

标准LMS算法的执行流程：

1) 滤波器初始化： $\mathbf{w}(0)$ 、$\mathbf{x}(0)$

2) 对每一个新的输入采样 $x(n)$ ，计算输出信号 $y(n)$

3) 利用期望输出 $d(n)$ ，根据等式(3.4)计算误差信号 $e(n)$ ，得到梯度(3.8)式。

4) 利用等式(3.9)更新滤波器系数：$\mathbf{w}(n)$

5) 返回步骤 2)，直至结束，可以得到输出序列和误差序列。

3.1.2 基本LMS算法

LMS算法的基本思想——梯度下降。

LMS算法的优缺点:

优点：算法简单、易实现。
缺点：收敛速度慢

LMS算法的改进思路：block LMS

梯度噪声

3.1.2 LMS的衍生算法

用更多的采样点，更新一次滤波器系数：

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu\mathbf{x}(n)e(n)$$

$$\mathbf{w}(n+2) = \mathbf{w}(n+1) + 2\mu\mathbf{x}(n+1)e(n+1)$$

…

$$\mathbf{w}(n+L) = \mathbf{w}(n+L-1) + 2\mu\mathbf{x}(n+L-1)e(n+L-1)$$

不难得出：

$$\boxed{\mathbf{w}(n+L) = \mathbf{w}(n) + 2\mu\sum_{m=0}^{L-1}\mathbf{x}(n+m)e(n+m)} \qquad (3.10)$$

时域LMS的分块更新(block recursion)公式

每 L 点更新一次

3.1.2 LMS的衍生算法

$$\mathbf{w}(n+L) = \mathbf{w}(n) + 2\mu \sum_{m=0}^{L-1} \mathbf{x}(n+m)e(n+m) \tag{3.10}$$

注意，(3.10)中的误差信号是通过同一个滤波器 $\mathbf{w}(n)$ 得到的：

$$e(n+m) = d(n+m) - \mathbf{x}^T(n+m)\mathbf{w}(n) \tag{3.11}$$

为了推导方便，我们通常令 $L = N$ 。

与标准LMS算法相比，*block LMS* 的更新频率降低了 *L* 倍，所以，我们定义一个新的时间索引 *k* 来代替 *n* 。

$$kL = n$$

3.1.2 LMS的衍生算法

$$\boxed{\mathbf{w}(n+L) = \mathbf{w}(n) + 2\mu \sum_{m=0}^{L-1} \mathbf{x}(n+m)e(n+m)} \qquad (3.10)$$

$$\Big\downarrow \; kL = n$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu \sum_{m=0}^{L-1} \mathbf{x}(kL+m)e(kL+m) \qquad (3.12)$$

其中将$\mathbf{w}(kL)$和 $\mathbf{w}(kL+1)$ 分别简写成$\mathbf{w}(k)$ 和$\mathbf{w}(k+1)$ 。所以，(3.12)和(3.10)是完全等价的。

由(3.12)式，可以直接写出 *block LMS* 的梯度，它其实是block中每个点的梯度求和：

$$\hat{\nabla}(k) = -2 \sum_{m=0}^{L-1} \mathbf{x}(kL+m)e(kL+m) \qquad (3.13)$$

3.1.2 LMS的衍生算法

block LMS的两个核心运算：

$$y(n+m) = \mathbf{x}^T(n+m)\mathbf{w}(n)$$

输入向量与滤波器系数向量的线性卷积

$$\hat{\nabla}(k) = -2\sum_{m=0}^{L-1}\mathbf{x}(kL+m)e(kL+m)$$

误差信号与输入向量的线性相关

（忽略常数项）

3.1.2 LMS的衍生算法

回顾线性卷积（相关）与圆周卷积（相关）的关系：

一般的，如果两个有限长序列的长度为 N_1 和 N_2，且满足 $N_1 \geqslant N_2$，则有圆周卷积的后 $N_1 - N_2 + 1$ 个点，与线性卷积的结果一致。

一般的，如果两个有限长序列的长度为 N_1 和 N_2，且满足 $N_1 \geqslant N_2$，则有圆周相关的前 $N_1 - N_2 + 1$ 个点，与线性相关的结果一致。

3.1.2 LMS的衍生算法

第一步：计算线性卷积

$$y(n+m) = \mathbf{x}^T(n+m)\mathbf{w}(n)$$

利用FFT计算线性卷积的有效方法：

overlap-save method

overlap-add method

Overlap-save method

为了得到 N 点的（线性卷积后的）输出信号：

$$y(n+m) = \mathbf{x}^T(n+m)\mathbf{w}(n) \qquad m = 0, 1, ..., N-1$$

要保证至少有 N 个点的线性卷积和圆周卷积的结果相同：

$$N_1 - N_2 + 1 = N$$

由于 $N_1 \geqslant N_2$（也就是输入信号长度通常大于滤波器的阶数），且 $N_2 = N$ (滤波器阶数为N)，那么要求每次参与运算的输入信号长度 N_1 至少为 $2N-1$

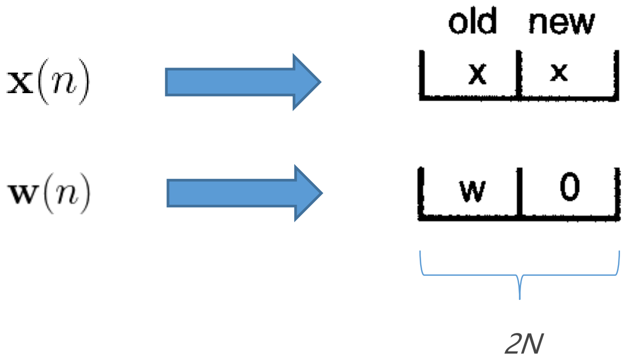
为了计算FFT方便，我们令输入信号的长度：

$$N_1 = 2N$$

显然，FFT的长度为2N

Overlap-save method

如何构造长度为2*N*的数据?



2 blocks，each block contains N data samples.

每个*N*阶滤波器后面补*N*个零.

Overlap-save method

只要理解了overlap-save的原理，下面的公式看起来就很容易了：

分别计算输入信号向量与滤波器系数向量的傅里叶变换：

$$\mathbf{X}(k) = diag\left\{\mathbf{F}\left[x(kN-N),\ \ldots, x(kN-1),\ x(kN),\ \ldots, x(kN+N-1)\right)\right]\right\} \tag{3.14}$$

$$\mathbf{W}(k) = \mathbf{F}\left[\mathbf{w}^T(k),\ 0,\ \ldots, 0\right]^T \tag{3.15}$$

频域里相乘：

$$\mathbf{Y}(k) = \mathbf{X}(k)\mathbf{W}(k) \tag{3.16}$$

则，*N*点线性卷积输出信号 $\mathbf{y}(k) = [y(kN),\ y(kN+1),\ \ldots, y(kN+N-1))]$ ，就等于 $\mathbf{Y}(k)$ 的傅里叶逆变换的后*N*个点：

$$\mathbf{y}(k) = last\ N\ samples\ of\ \mathbf{F}^{-1}\mathbf{Y}(k) \tag{3.17}$$

Overlap-save method

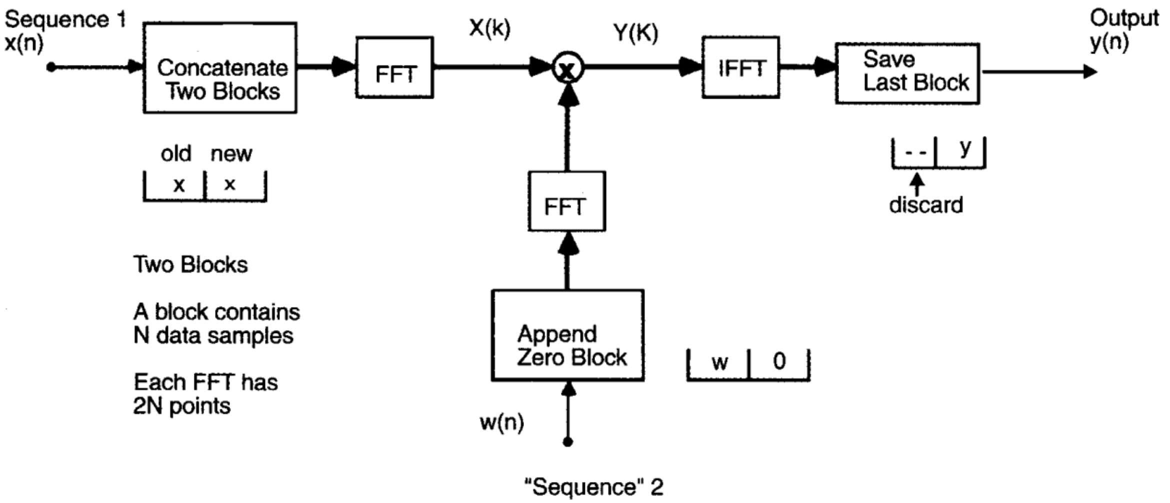


Figure: overlap-save sectioning method

注意，overlap-save方法不仅适用于频域自适应滤波，对于滤波器系数固定不变的情况依然适用。例如我们常见的语音加混响，其实就是计算纯净语音信号，与固定系数的房间冲激响应之间的线性卷积，可以使用overlap-save方法实现。

梯度的估计

第二步：解决线性相关

$$\hat{\nabla}(k) = -2\sum_{m=0}^{L-1}\mathbf{x}(kL+m)e(kL+m)$$

与对卷积的操作非常类似，为了计算输入向量与误差信号之间的线性相关，我们的思路仍然是想办法将它们变换到频域，计算输入信号的共轭谱与误差信号谱的乘积。

输入信号 $\mathbf{X}(k)$ 在上一步处理卷积运算是已经求得。

那么，剩下的工作就是，将误差向量 $\mathbf{e}(k) = [e(kN),\ e(kN+1), \ldots, e(kN+N-1)]$ 也扩展到 *2N* 长度：

梯度的估计

计算卷积的时候，我们将滤波器系数向量后面补N个零。

那么，计算相关的时候，我们在误差向量前面补N个零。（大家想一想其中的原因）

补零之后，计算误差向量的傅里叶变换：

$$\mathbf{E}(k)=\mathbf{F}\left[0,\ 0,\ \ldots,0,\ \mathbf{e}^{T}(k)\right]^{T} \qquad (3.18)$$

与卷积运算对比 对比(3.15)

频域里相乘：

$$\mathbf{\Delta}(k)=\mathbf{X}^{H}(k)\mathbf{E}(k) \qquad (3.19)$$

对比(3.16)

则，N点时域梯度向量 $\left[\hat{\bigtriangledown}(kN),\ \hat{\bigtriangledown}(kN+1),\ \ldots,\hat{\bigtriangledown}(kN+N-1)\right]$，就等于 $\mathbf{\Delta}(k)$ 的傅里叶逆变换的前N个点：

$$\vec{\bigtriangledown}(k)=first\ N\ samples\ of\ \mathbf{F}^{-1}\mathbf{\Delta}(k) \qquad (3.20)$$

对比(3.17)

frequency-domain adaptive filtering (FDAF)

第三步：滤波器系数更新

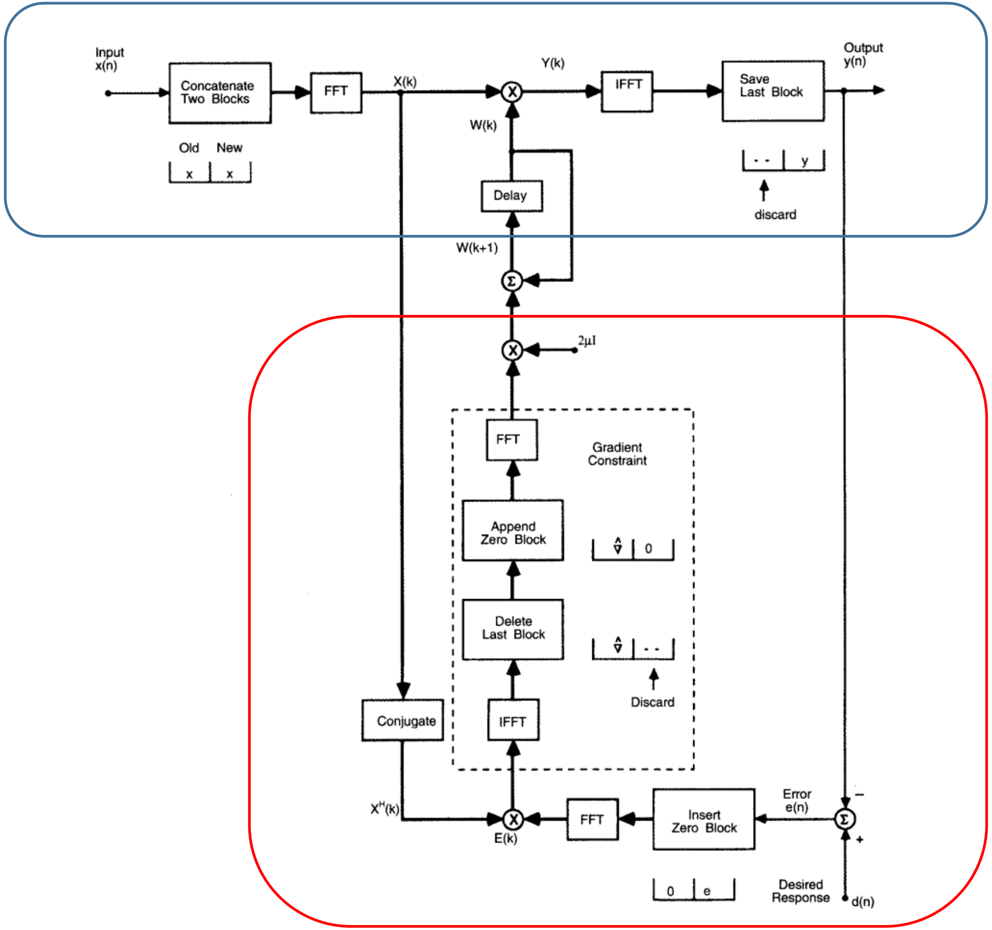
$$\mathbf{W}(k+1) = \mathbf{W}(k)+2\mu\mathbf{F}\left[\vec{\nabla}^{T}(k),\ 0,\ \ 0,\ \ldots,0\right]^{T} \qquad (3.21)$$

注意：

第一，滤波器系数直接在频域更新，所以需要将梯度向量再次变换到频域；

第二，由于滤波器系数向量后面补了N个零【参考等式(3.15)】，为了保证结果的正确性，梯度向量也需要在后面补N个零。

frequency-domain adaptive filtering (FDAF)



计算输出（卷积）

计算梯度（相关）

关于LMS算法的几点讨论

Q1：LMS算法对输入信号有何要求？

$$\mathbf{w}(n+1)=\mathbf{w}(n)+2\mu\mathbf{x}(n)e(n) \tag{3.9}$$

LMS要求不同时刻的输入向量 $\mathbf{x}(n)$ 线性无关——LMS的独立性假设。如果输入信号存在相关性，会导致前一次迭代产生的梯度噪声传播到下一次迭代，造成误差的反复传播，收敛速度变慢，跟踪性能变差。

所以，理论上，LMS算法对白噪声的效果最好。

为了降低输入信号的相关性，出现了一类“解相关LMS”算法，这里就不展开讲述了。

关于LMS算法的几点讨论

Q2：何为归一化LMS、功率归一化LMS？

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu \frac{\mathbf{x}(n)}{\mathbf{x}^T(n)\mathbf{x}(n)} e(n) \qquad (3.22)$$

(3.22)式为归一化LMS（NLMS）算法的更新公式。与标准LMS算法相比，梯度项中增加了一个归一化项：$\mathbf{x}^T(n)\mathbf{x}(n)$

归一化的作用：对于较大的输入，会导致梯度噪声的放大，因此需要用输入向量的平方范数进行归一化。

关于LMS算法的几点讨论

Q2：何为归一化LMS、功率归一化LMS？

$$\mu(n) = \frac{\alpha}{\beta + \mathbf{x}^T(n)\mathbf{x}(n)}, \quad 0 < \alpha < 2, \quad \beta \geqslant 0 \qquad (3.23)$$

功率归一化LMS（PNLMS）是利用归一化项 $\mathbf{x}^T(n)\mathbf{x}(n)$ 调整学习速率，数值稳定性要好于NLMS。当 $\alpha = 1$ 时，PNLMS退化为NLMS。

关于LMS算法的几点讨论

Q3：学习速率如何选取？

$$\boxed{\mathbf{W}(k+1) = \mathbf{W}(k)+2\mu\mathbf{F}\left[\vec{\nabla}^T(k),\ 0,\ \ 0,\ \ldots,0\right]^T} \tag{3.21}$$

我们在推导频域自适应滤波方法的时候，为了简化问题，将每一个frequency bin上的学习速率均设为常数 μ 。

在实际工程应用中，用的更多的一种方法是，对第m个frequency bin，利用输入信号在这个频点的功率 $P_m(k)$ 对学习速率进行归一化：

$$\mu_m(k) = \frac{\mu}{P_m(k)} \tag{3.22}$$

频点功率 $P_m(k)$ 通常采用迭代的方式求得：

$$P_m(k) = \lambda P_m(k-1) + (1-\lambda)\left|X_m(k)\right|^2 \tag{3.23}$$

本章回顾

3.1 LMS算法

3.1.1 自适应滤波器的应用

3.1.2 LMS算法

3.2 实战

Overlap-save算法实现

请大家实现一个给纯净语音加混响的函数：
int conv(short* inputdata, long inputdata_length, double* rir, long rir_length, short* outputdata);

输入：inputdata：纯净语音数据
inputdata_length：语音长度
rir：房间冲激响应函数
rir_length：冲激响应函数的阶数

输出：outputdata：加混响后的语音
方法：overlap-save 算法
语言：C/C++

函数的调用，外层的数据准备，我们都已经写好，请大家完成conv这个核心函数，并观察处理结果是否合理。

感谢聆听！

Thanks for Listening